REPORT NO. GDC-DDF67-004

Applications of

DISCRETE NETWORK SIMULATION IN SPACE VEHICLE CHECKOUT FINAL REPORT VOLUME II COMPUTER PROGRAMS

GENERAL DYNAMICS
Convair Division

REPORT NO. GDC-DDF67-004

Applications of

DISCRETE NETWORK SIMULATION IN SPACE VEHICLE CHECKOUT FINAL REPORT VOLUME II COMPUTER PROGRAMS

DECEMBER 1967

Submitted to
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA
under
CONTRACT NAS8-21102

Prepared by
CONVAIR DIVISION OF GENERAL DYNAMICS
HUNTSVILLE, ALABAMA

TABLE OF CONTENTS

ILLUSTRATIONS

PRECEDING PAGE BLANK NOT FILMED.

SUMMARY

The family of Discrete Network Simulation Programs developed by the Convair division of General Dynamics were initially developed as a tool for time oriented simulation and analysis of man machine systems. With this technique the operation of complex systems can be accurately simulated. During the development of DNS, repeated testing has proven its value as an aid to technical data generation, projected failure analysis, and automatic malfunction analysis. It has been shown to be a versatile tool thats application is limited only by the imagination of the user. Under the present contract the application of DNS as an instrument to aid in the validation of Saturn V system test procedures was initiated.

In order to optimize the technique of that procedure validation, and effect on overall improvement in the versatility of DNS, certain programming tasks were undertaken during this contract period. Changes were incorporated to existing DNS programs and several new programs were written to accomplish the task of test procedure validation. The new programs are covered in detail in this volume.

INTRODUCTION

A test procedure may be simulated and validated by stimulating a model of the system in accordance with the test procedure, and comparing the results of the simulation with the predicted test results step by step. Figure 1 illustrates the relationships between the DNS/Test Procedure Validation programs. The following techniques and programs are employed in the validation process:

SYSTEM DESCRIPTION

Boolean equations describing the system to be modeled are punched on cards and assembled into an Equation Card Deck.

TIME CARD GENERATOR PROGRAM

The Equation Card Deck is then processed by the NAME/TIME Card Generator Program which compiles a list of variables from the equations, determines the required timing data for each variable in the list, and punches this information on cards to create the Time Card Section for the model.

UPDATE PROGRAM

The Time Card Deck and Equation Card Deck are formed into a Master Data File, and stored on tape using the Update Program. The printout is thoroughly analyzed and changes or corrections are inserted into the Master Data File again using the Update Program.

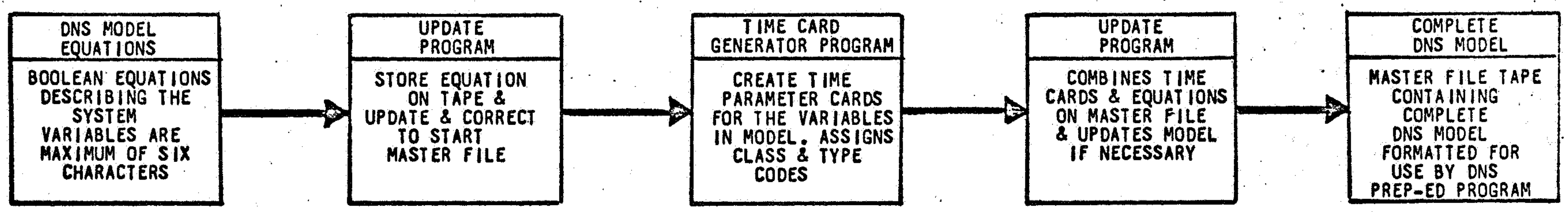
PREPROCESSOR-EDITOR PROGRAM

The Preprocessor-Editor Program[1] converts and cross references the system description from the Master Data File, and stores this information on the model tape formatted for use by the Simulation Program.

INPUT CONVERSION PROGRAM

The Input Conversion and Punch Program generates the driving functions required by the Simulation Program directly from the ATOLL card image tape for the test procedure. The program may be used with either Boeing or IBM ATOLL formats.

DISCRETE NETWORK SYSTEM MODEL CONSTRUCTION


DNS MODEL EQUATIONS
BOOLEAN EQUATIONS DESCRIBING THE SYSTEM VARIABLES ARE MAXIMUM OF SIX CHARACTERS
UPDATE PROGRAM
STORE EQUATION ON TAPE & UPDATE & CORRECT TO START MASTER FILE
TIME CARD GENERATOR PROGRAM
CREATE TIME PARAMETER CARDS FOR THE VARIABLES IN MODEL. ASSIGNS CLASS & TYPE CODES
UPDATE PROGRAM
COMBINES TIME CARDS & EQUATIONS ON MASTER FILE & UPDATES MODEL IF NECESSARY
COMPLETE DNS MODEL
MASTER FILE TAPE CONTAINING COMPLETE DNS MODEL FORMATTED FOR USE BY DNS PREP-ED PROGRAM

TEST PROCEDURE VALIDATION

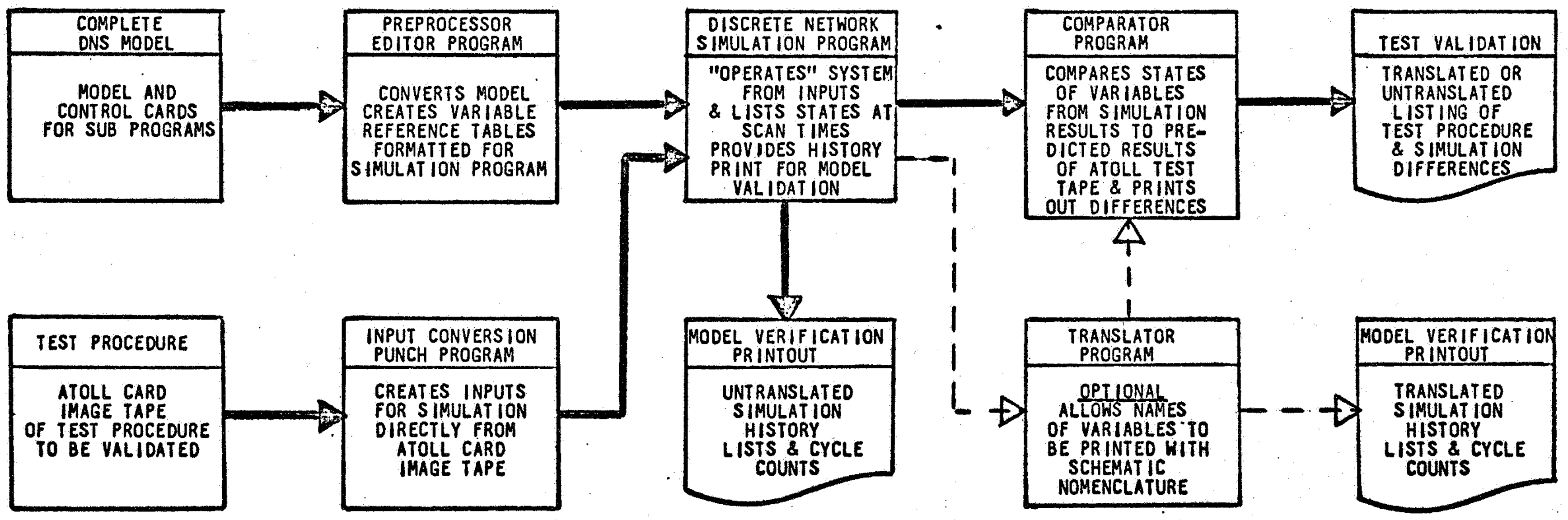

COMPLETE DNS MODEL
MODEL AND CONTROL CARDS FOR SUB PROGRAMS
PREPROCESSOR EDITOR PROGRAM
CONVERTS MODEL CREATES VARIABLE REFERENCE TABLES FORMATTED FOR SIMULATION PROGRAM
DISCRETE NETWORK SIMULATION PROGRAM
"OPERATES" SYSTEM FROM INPUTS & LISTS STATES AT SCAN TIMES PROVIDES HISTORY PRINT FOR MODEL VALIDATION
COMPARATOR PROGRAM
COMPARES STATES OF VARIABLES FROM SIMULATION RESULTS TO PRE-DICTED RESULTS OF ATOLL TEST TAPE & PRINTS OUT DIFFERENCES
TEST VALIDATION
TRANSLATED OR UNTRANSLATED LISTING OF TEST PROCEDURE & SIMULATION DIFFERENCES
TEST PROCEDURE
ATOLL CARD IMAGE TAPE OF TEST PROCEDURE TO BE VALIDATED
INPUT CONVERSION PUNCH PROGRAM
CREATES INPUTS FOR SIMULATION DIRECTLY FROM ATOLL CARD IMAGE TAPE
MODEL VERIFICATION PRINTOUT
UNTRANSLATED SIMULATION HISTORY LISTS & CYCLE COUNTS
TRANSLATOR PROGRAM
OPTIONAL ALLOWS NAMES OF VARIABLES TO BE PRINTED WITH SCHEMATIC NOMENCLATURE
MODEL VERIFICATION PRINTOUT
TRANSLATED SIMULATION HISTORY LISTS & CYCLE COUNTS

Figure 1. TEST PROCEDURE VALIDATION TECHNIQUE.

These driving functions consist of punched and sequenced input commands and control cards.

SIMULATION PROGRAM

The Simulation Program[1] uses the driving functions to stimulate the model and control the simulation in accordance with the test procedure. The results of the simulation are stored on a simulation output tape. As each variable changes state, a cycle counter is incremented by one, providing a record of the number of changes of state of each variable for a given test. Each input is analyzed for its effect on the system, and the resulting history of inputs and reactions are created for all the variables. Printout of this 'event trail' is optional to the extent that all, none, or only selected portions may be printed.

TRANSLATOR PROGRAM

The Translator Program substitutes descriptions obtained from an input dictionary for the coded names in the output from a test procedure simulation. The program allows the simulation history and the comparator listings to be printed with the variables identified by hardware nomenclature. The nomenclature can be referrable directly to system schematics if desired.

COMPARATOR PROGRAM

The Comparator Program validates the test procedure by comparing the results of the test procedure simulation with the ATOLL predictions for the test procedure, and lists any differences encountered. Areas of differences are manually examined to determine the reason for the difference and will fall into one of these three categories.

1. Error in ATOLL tape.

2. Error in schematics.

3. Error in DNS model.

AUTOMATIC MALFUNCTION ANALYSIS PROGRAMS[1]

The Input Conversion and Punch Program, and the Comparator Program were written for test procedure validation exclusively during the period of this contract. The Time Card Generator, Translator Program, and the Update Program, were written as DNS model building improvements during the period of this contract, and may be used for AMA application as well. The following modifications were made to the Simulation Program to improve the test procedure validation technique.

1. Provide increased data handling capability. This was required to permit the complete test procedure model to be simulated normally without

exceeding the program data storage limits available during the runs. The additional data storage area was obtained by removing all references and routines pertaining to the conversion and processing of the Binary Simulation Output Tape, and substituting a printed output save tape for subsequent use with the Comparator Program. This modification permitted increasing the data limits by two thousand IBM words.

2. Provide for labeling in the simulation history any input variable whose state in the value table is already at the value requested. This permits identification of any test procedure commands which may be redundant.

3. Provide routines to count and store the number of times a variable undergoes a change of state (cycles) during the course of a test procedure simulation. The cycle counting is activated when a *HEAD LIST control card is encountered, and the current cycle count for each variable is then printed whenever a state list is requested. This permits identifying variables which may be being overworked, and in conjunction with the Comparator Program, permits listing all variables which were not used at all during a test procedure.

The Down Translation and Culling Program (DTC) is not required for test procedure validation if the variable name length is held to a maximum of six characters. If expanded names are desired for test procedure validation, the Translation Program will provide the uptranslated name when used in the process depicted by Figure 1. Modifications to the DTC Program were incorporated to expand its overall efficiency for automatic malfunction analysis technique usage. Modifications were as follows:

1. Provide the capability to accept input data from either the time and equation card deck or the basic data tape from the Update Program.

2. Assign the three character code names to inactive variables for inclusion in the printed output.

3. Provide segregation of processed equation data printouts by equation numbers.

4. Provide the capability to read formatted time card names.

A representative printout from the modified program is shown in Figure 2.

As with the DTC Program the REFTAB (Variable Reference Table) step is not utilized during test procedure validation but was designed specifically to assist in verifying the accuracy and completeness of a modelled system intended for use with AMA. This program reads the AMA model tape previously created by the DT&C Program, converts the binary coded data describing each variable, and prints out a reference data list which summarizes the identifications, classification, type, and use of each variable in the AMA model.

```
                        DISCRETE NETWØRK SIMULATØR-DØWNTRANSLATED TIME CARDS

AAA   DI1345                                      S   0S   0S   0S   0S   0S   0AIFU  I0
AAB   NØDE115A3A1J34P                             S   0S   0S   0S   0S   0S   0AD X  X0
AAC   PBUS5A31D111                                S   0S   0S   0S   0S   0S   0AI I  B0
AAD   PBUS115A3A21D111                            S   0S   0S   0S   0S   0S   0AI I  B0
AAE   CØIL115A3A1K40J15PPSZ                       S   0S   0S   0S   0S   0S   0AIFU  K0
AAF   CØNT115A3A1K40J15AABB                       S   4S   5S   6S   4S   5S   6ADUM  C0
AAG   CØNT115A3A2K51J20DDEE                       S   4S   5S   6S   4S   5S   6ADUM  C0
AAH   PIN5A3J34B                                  S   0S   0S   0S   0S   0S   0A     P1
AAI   CK1N115A3A1J34P                             S   0S   0S   0S   0S   0S   0AIUS  Y0
AAJ   CK2N115A3A1J34P                             S   0S   0S   0S   0S   0S   0AIUS  Y0

                                                    INACTIVE VARIABLES

AAW         CØNT115A3A2K50J30DDEE                           IDUM  C1
AAX         PIN5A3J34B                                      I     P1
AAY         PIN115A3A1J20C                                  I     P1
AAZ         CK3N115A3A1J34P                                 IIUS  Y0
ABA         USEDINAC-ZERØ                                   I  I  G1
ABB         UNUSEDINAC-BØTH                                 IDUM  C1

                                                         UNSPECIFIED NAMES LIST
EQUATIØN              NAME

      1     CK4N115A3A1J34P

      2     PIN115A3A1J16A
      2     PIN115A3A2J21B
      2     PIN115A3A2J16B

      3     PIN115A3A1J34B

      4     PIN115A3A1J34C
```

Figure 2. Printout of revised DTC program (1 of 2).

```
                DISCRETE NETWØRK SIMULATØR - DØWNTRANSLATIØN ØUTPUT
 *-ØRIGINAL EQUATIØN  **-CULLED EQUATIØN ***-TRANSLATED AND CULLED EQUATIØN      PAGE   1

  *   NØDE  115A3 A1        J34 P     =   CK1N  115A3 A1         J34 P     +  EQ 1    1   1
  *                                       CK2N  115A3 A1         J34 P     +  EQ 1    2   1
  *                                       CK3N  115A3 A1         J34 P     +  EQ 1    3   1
  *                                       CK4N  115A3 A1         J34 P     .  EQ 1    4   1
 **   NØDE115A3A1J34P=CK1N115A3A1J34P+CK2N115A3A1J34P.                        EQ 1    1   1
***    AAB  = AAI  + AAJ  .                                                   EQ 1    1   1

  *   CK1N  115A3 A1        J34 P     = / PIN   115A3 A1         J20 C     *  EQ 2    1   2
  *                                       CØNT  115A3 A1  K40    J15 AABB  *  EQ 2    2   2
  *                                     / PIN   115A3 A1         J16 A     *  EQ 2    3   2
  *                                     / PIN   115A3 A2         J21 B     *  EQ 2    4   2
  *                                     / CØNT  115A3 A2  K50    J30 CDEE  *  EQ 2    5   2
  *                                     / PIN   115A3 A2         J16 B     *  EQ 2    6   2
  *                                       PBUS  115A3 A2 1D111             .  EQ 2    7   2
 **   CK1N115A3A1J34P=CØNT115A3A1K40J15AABB*PBUS115A3A21D111.                 EQ 2    1   2
***    AAI  = AAF  * AAD  .                                                   EQ 2    1   2

  *   CK2N  115A3 A1        J34 P     = / PIN   115A3 A1         J34 B     *  EQ 3    1   3
  *                                     / PIN   5A3              J34 B     *  EQ 3    2   3
  *                                       PRSW  5A3       50A45C           *  EQ 3    3   3
  *                                       PBUS  5A3       1D111            .  EQ 3    4   3
 **   CK2N115A3A1J34P=/PIN5A3J34B*PRSW5A350A45C*PBUS5A31D111.                 EQ 3    1   3
***    AAJ  =/ AAH  * AAK  * AAC  .                                           EQ 3    1   3

  *   CK3N  115A3 A1        J34 P     = / PIN   115A3 A1         J34 C     *  EQ 4    1   4
  *                                       CØNT  115A3 A2  K51    J20 CDEE  *  EQ 4    2   4
  *                                       PBUS  115A3 A2  1D111            .  EQ 4    3   4

  *   CØNT  115A3 A1  K40   J15 AABB  =   CØIL  115A3 A1  K40    J15 PPSZ  .  EQ 6    1   5
 **   CØNT115A3A1K40J15AABB=CØIL115A3A1K40J15PPSZ.                            EQ 6    1   5
***    AAF  = AAE  .                                                          EQ 6    1   5
```

Figure 2. Printout of revised DTC program (2 of 2).

SECTION

1

DNS UPDATE PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

1

DNS UPDATE PROGRAM

AUTHOR: **T. C. Larson**
Convair division of General Dynamics

PURPOSE: The Discrete Network Simulation (DNS) Program requires the use of large quantities of IBM cards to describe a large network or system. During the "model building" phase it is necessary to load these cards into the computer each time a change (correction or system configuration alteration) is incorporated into the model.

The Update Program provides DNS programs with a model on tape that may be readily changed with a minimum of card handling.

The function of the Update Program is to:

1. Generate an initial model tape with each card image assigned an ascending sequence number, in a format compatible with subsequent DNS programs.

2. Insert new (additional) card images into model where directed.

3. Delete cards from model as directed.

4. Correct individual cards as directed.

5. Generate a model tape with new sequence numbers.

This program was developed originally for IBM 7090/94 use but is written entirely in Fortran IV for compatibility with other data processing systems.

RESTRICTIONS: If program sorting of the input data is requested, total data deck input is limited to 702 cards (including control cards).

STORAGE: This program, including files, occupies $31{,}327)_8$ consecutive locations in 7090/94 memory. The first subroutine in the object program 'ASETUP' starts at location 03047 and the final subroutine 'ENDEM' fills memory through location 20670 - a total of $15{,}621)_8$ consecutive locations.

	Subroutine	Function
1.	ASETUP	Driver
2.	CD2BIN	BCD to binary conversion
3.	CUPDAT	Read and write tapes and generate sequence numbers
4.	CONCRX	Identification and interpretation of control cards
5.	GALOAD	Initial tape and sequence number generation
6.	HERROR	Printing of error messages
7.	RDTCD	Card reading and end of data sensing
8.	TITLEX	Paging and title printout
9.	UPSRX	Stores data and generates sorted control words for sequential card use
10.	ENDEM	Write tape EOF, rewind tapes, and prints new tapes

TIMING: Output of the program is approximately 1100 lines per minute. Thus, a tape containing 4500 records could have approximately 200 records updated in 4 minutes, including complete listings.

This time could be reduced slightly by writing the print tape as the model is updated, but for this particular DNS application it was ascertained that it would be of more value to the user if the 'save' tape is rewound and printed, providing the user with an actual printout of the DNS model input tape.

USE: The program operates in three modes. Tape requirements are listed for each mode. The first data card contains all necessary control words for program mode selection.

Mode 1

Col. 1-12	must have *LOADbCARDSb
Col. 13-66	blank
Col. 67-80	any information pertinent to tape ID, (may be left blank)

This mode reads the model from cards, stores the card image, generates sequence numbers, and writes the card images and sequence numbers on a tape. At completion of program, it rewinds the tape and provides a printout for reference.

Tape Requirements -

Fortran Logical	System Function	Tape
8	A5	DNS Model

Mode 2

Col. 1-7	must contain *UPDATE
Col. 8-66	blank
Col. 67-80	(optional) same as Mode 1

This mode assumes that new cards are pre-sorted in ascending sequence numbers. Out of sequence cards are ignored and will not be included in the new tape. They will be printed with notation to this effect.

Tape Requirements -

Fortran Logical	System Function	Tape
8	A5	Existing DNS Model
12	A7	Updated DNS Model

Mode 3

Col. 1-7	must contain *UPDATE
Col. 8-24	blank
Col. 25-30	*SORTb
Col. 31-66	blank
Col. 67-80	(optional) same as Mode 1

This mode assumes that update cards are not pre-sorted in ascending sequence numbers and processes all cards accordingly. It should always be used when numerous corrections are made to the model.

Tape Requirements - Same as Mode 2

For Modes 2 and 3 the next sequential data card must be one or more of the three $ control cards listed.

$ADD

Col. 1-6	$ADDbb
Col. 7-24	blank
Col. 25-30	The sequence number on existing tape where new cards are to be added. Right adjusted to Col. 30.
Col. 31-62	blank

This card causes all cards immediately following up to next '$' control card to be inserted into the model at the point designated in Col. 25-30. Card columns between Col. 25-30 that do not contain numbers must contain blanks.

$DELETE

Col. 1-7	$DELETE
Col. 8-24	blank
Col. 25-30	The sequence number of the first card in existing model to be deleted. Right adjusted to Col. 30.
Col. 31-36	blank or word thru
Col. 37-42	The sequence number of the last card in existing model to be deleted. Right adjusted to Col. 42.
Col. 43-80	blank

If only one card is being deleted, Col. 37-42 may be left blank or the number of the deleted card may be repeated. Card columns between 25-30 and 37-42 not containing numbers must be left blank.

$CORRECT

Col. 1-8	$CORRECT
Col. 9-24	blank
Col. 25-30	The sequence number of the card in existing model to be changed. Right adjusted to Col. 30.

Card columns between 25-30 not containing numbers must be blank. Card immediately following is card that will replace the existing card indicated in card Cols. 25-30.

The following card is required in all modes and is always the last data card in the data deck.

*END DATA

Col. 1-12	must have *ENDbDATAbbb
Col. 13-80	blank

DECK SET UP:

A typical deck set up applicable to Mode 1 is as follows:

1. Binary program deck.
   $DATA

2. *LOAD CARDS (optional tape ID)
   -
   -

3. (Data to be loaded)
   -
   -

4. *END DATA

5. End of File (EOF) (7-8 punch)

A typical deck set up for Modes 2 and 3 is as outlined below:

1. Binary program deck.
   $DATA

2. *UPDATE   *SORT (or blank)   (optional tape ID)

3. $ADD   5266 (old tape sequence number)

4. -
   (Cards to be loaded)
   -

5. $DELETE   350 through 520

6. $CORRECT   1501

7. (Card to be inserted in place of one listed)

8. *END DATA

9. EOF

METHOD: The course followed throughout the program is:

1. Set up counters, flags, and control parameters and ready tapes.

2. Read first control card.

3. Determine mode of operation and set flags.

4. If 'load cards' mode is selected, program follows steps A through H.

   A. Reads data cards one at a time.

   B. Assigns sequence number.

   C. Writes on save tape.

   D. When end of data is sensed, writes *END DATA and EOF on save tape.

   E. Rewinds save tape.

   F. Reads save tape and writes on print tape.

   G. Rewinds save tape.

   H. End run.

5. If UPDATE mode is selected, program follows steps I through Q.

   I. Reads data card and determines classification i.e., control card or input card.

   J. If control card directs processing to appropriate portion of program, add, delete, or correct. Obtains point in existing tape where UPDATE is to be inserted.

   K. If input card processes card through portion of program as directed in step J. Process includes reading existing tape up to point directed. As the old tape is read, card images are rewritten with new sequence numbers on new save tape.

L. When addition or correction point is reached, new card/cards are then added to new tape with continuing sequence numbers.

M. Cards added or deleted are printed out for reference. In the case of corrections, both old card and new card is printed.

N. When end of data deck is sensed, remainder of old tape is read and rewritten on new tape with new sequencing until tape end of data is encountered.

O. If during processing an out of sequence insertion number is encountered, the control card and its associated cards are not processed but flagged and printed for reference.

P. Program transfers to processing as previously stated in steps D through G.

Q. End run.

6. If update sort option is selected, all data cards are read into memory and processed as in steps R through Z.

R. As each card is read, it is classified as a control or input card.

S. As each control card is encountered, a 3 word record is generated.

Word 1	Contains the sequence number listed on the control card.
Word 2	Contains the number of the core storage cell where the first word of the control card is stored.
Word 3	Contains the number of words stored pertaining to this control card. This includes the number of words in the control card, plus the number of words in all associated cards that follow.

T. When the end of data is sensed, it is assigned a number large enough to ensure that it will be the largest sequence number.

U. The three word records are then sorted by the first word (sequence insertion numbers) in ascending order.

V. A flag is set that will direct other portions of the program to obtain data from core instead of input tape.

W. Program then transfers the card images into the update portion of the program as outlined in steps X through Y.

X. The location of the card in memory is obtained from the second word of the sorted three word record. The number in word three is noted and as the card image is transferred to the main program for processing, it is reduced by the number of words transferred.

Y. After processing each card image, the previous number noted from word three is checked for a zero value. If it is not zero, the next card image is obtained from the sequence core location of the previous card processed. If it is zero, the next three word record is obtained and a new core location and word count is noted for the next card.

Z. Processing of cards and program termination is identical as explained in steps J through Q.

OUTPUT FORMAT:

A sample of the DNS Update Program is illustrated in Figures 1-1 thru 1-3. The corrections that were made to the model tape will be listed starting on Page 1 of the printout. Figure 1-1 is a composite sample taken from an actual update test run. The information (SORT TEST) printed on the title line after "Tape ID" is an optional input for identifying the model tape. As the update of the tape processes, the changes are listed as they occur. The sequence number appearing at the right of the cards that were added is the new sequence number and will not necessarily agree with the number on the update $ADD card.

The cards deleted in the example have the old tape sequence number printed on the right. Correction cards will have both the old card and new card listed in that order, including old and new sequence numbers. The error in the example shown is underlined and the correct card has the relay number changed to 28K60.

```
TAPE ID.  SØRT TEST                 DESCRETE NETWØRK UPDATE PRØGRAM                    PAGE  1

                        THE FØLLØWING CARDS WERE ADDED

DI423      S    OS   OS   0    S   OS   OS   0                                          253
DI424      S    OS   OS   0    S   OS   OS   0                                          254
DI425      S    OS   OS   0    S   OS   OS   0                                          255
DI426      S    CS   OS   0    S   OS   OS   0                                          256
DI427      S    OS   OS   0    S   OS   OS   0                                          257
DI428      S    OS   OS   0    S   OS   OS   0                                          258
DI429      S    OS   OS   0    S   OS   OS   0                                          259

                        THE FØLLØWING CARDS WERE DELETED

241KA     = 241KA */242KB * 6D110 + 241KA * 241K2A */241K2B * 6D110 +    241 A  1      3805
            241K1A * 6D11C.                                              241 A  2      3806
241K1A    = 241K1B * 241K2B * 6D110 + 241K1B */241K5 * 6D110 +          241 B  1      3807
            241K1B */242KD * 6D110.                                      241 B  2      3808
241K1B    =DØ972.                                                        241           3809
241K2A    =DØ973.                                                        241           3810
241K2B    =98K115 *6D110.                                                241           3811
241K5     =DØ976.                                                        241           3812
242K3     = DØ974.                                                       242           3813
242K4     = DØ975.                                                       242           3814
242KB     = /242KD */242KC */241K1A * 6D110.                             242           3815
242KC     = /242K3 * 241KA * 241K1A * 6D110 +/242K3 * 241KA *           242 A  1      3816
            241K1A */242KB * 6D110 +/242K3 * 241KA * 241K1A *           242 A  2      3817
            241K2A */241K2B * 6D110.                                     242 A  3      3818

                        THE FØLLØWING CARD WAS CØRRECTED

L1N28A    =DØ195 *6D191 *28K60 +6D92 *DPLUG *28K40.                      28            4456
L1N28A    =CØ195 *6D191 *28K6C +6D92 *DPLUG *28K60.                      28            4504
```

Figure 1-1. New card listing from update program.

If the *SORT mode was not used and a control card was entered out of sequence, the corrections will not be incorporated in the new tape, and the cards will be printed flagged by asterisks as shown in the example on Figure 1-2.

After all corrections have been incorporated, the SAVE tape is rewound and a print of the new tape made. This consists of the time cards and equations as shown in Figure 1-3.

```
TAPE ID.  SEQ TEST                    DESCRETE NETWØRK UPDATE PRØGRAM                    PAGE   2


                    THE FØLLØWING CARDS WERE ADDED

L1NC7A   =6D110 *C7KA *(/C7K2A *C7K2B +C7K2A */C7KE +/C8KB).           C7           4428
L2NC7A   =C7K1A */C7K4B *C7K1B */C7K2A *6D110.                         C7           4429
L3NC7A   =C7K1A *6D110.                                                C7           4430

THE FØLLØWING CØNTRØL CARD AND/ØR CARDS WERE ØUT ØF SEQUENCE AND WERE NØT INCØRPØRATED

     ****  $CØRRECT                  30
     ****  30  C75K1      S   4S   5S    6    S    4S    5S    6

                    THE FØLLØWING CARD WAS CØRRECTED

L2N28A   =DØ195 *6D191 */28K60 *28K61.                                               4457
L2N28A   =DØ195 *6D191 */28K60 *28K61 +/28K60 *DPLUG *6D92 *28K61.    28            4505
*END DATA



TAPE ID.  SEQ TEST                    DESCRETE NETWØRK UPDATE PRØGRAM                    PAGE   3



*TIMES                                                                                  1
DI1        S    OS    OS    0    S    OS    OS    0                A                    2
DI2        S    OS    OS    0    S    OS    OS    0                                     3
DI3        S    OS    OS    0    S    OS    OS    0                A                    4
```

Figure 1-2. Out of sequence listing from update program.

```
TAPE ID.  SØRT TEST                    DESCRETE NETWØRK UPDATE PRØGRAM                PAGE  7

*TIMES                                                                                     1
DI1         S     OS     OS     0     S     OS     OS     0               A                2
DI2         S     OS     OS     0     S     OS     OS     0                                3
DI3         S     CS     OS     0     S     OS     OS     0               A                4
DI4         S     CS     OS     0     S     OS     OS     0                                5
DI5         S     OS     OS     0     S     OS     OS     0                                6
DI6               OS     OS     0     S     OS     OS     0                                7
D                               0     S     OS     OS                                      8
                                      S     OS     OS
            S                         S     OS
  D95       S     OS     OS
6L6D95      S     OS     OS     0                       0                               2367
7L6D95      S     OS     OS     0                 OS    0                               2368
8L6D95      S     OS     OS     0     S     OS    OS    0                               2369
9L6D95      S     OS     OS     0     S     OS    OS    0                               2370
*END TIMES                                                                              2371
*EQUATIØNS                                                                              2372
DI1      =31P28V *31K26 +31P28V *31PK1 *31K10 */31K8 */31K7 +31P28V *   31A    1        2373
          31PK1 *31K6 .                                                  31A    2        2374
DI2      =6D100A *39K12.                                                 31             2375
DI3      =31P28V *31K5.                                                  32             2376
DI4      =20D110 *42K14                                                  42             2377
DI5      =50K26 *                                                        55             2378
DI6      =33K                                8).                         33             2379
DI7                                        5PK1.                         35A    1        2380
                26K46 */26K48 *
          K46 */26K48 *N28B.
         =/27K43 */27K45 *N27B.                                                            1
8L6D95   =/27K44 */27K45 *N27B.                                                         4532
9L6D95   =/27K44 */27K43 *N27A.                                                         4533
*END EQUATIØNS                                                                          4534
*$$$$$
*END DATA       GENERAL DYNAMICS/CØNVAIR  HUNTSVILLE ØPERATIØNS
```

Figure 1-3. Composite master file tape listing of update program.

APPENDIX A

PROGRAM FLOW CHARTS

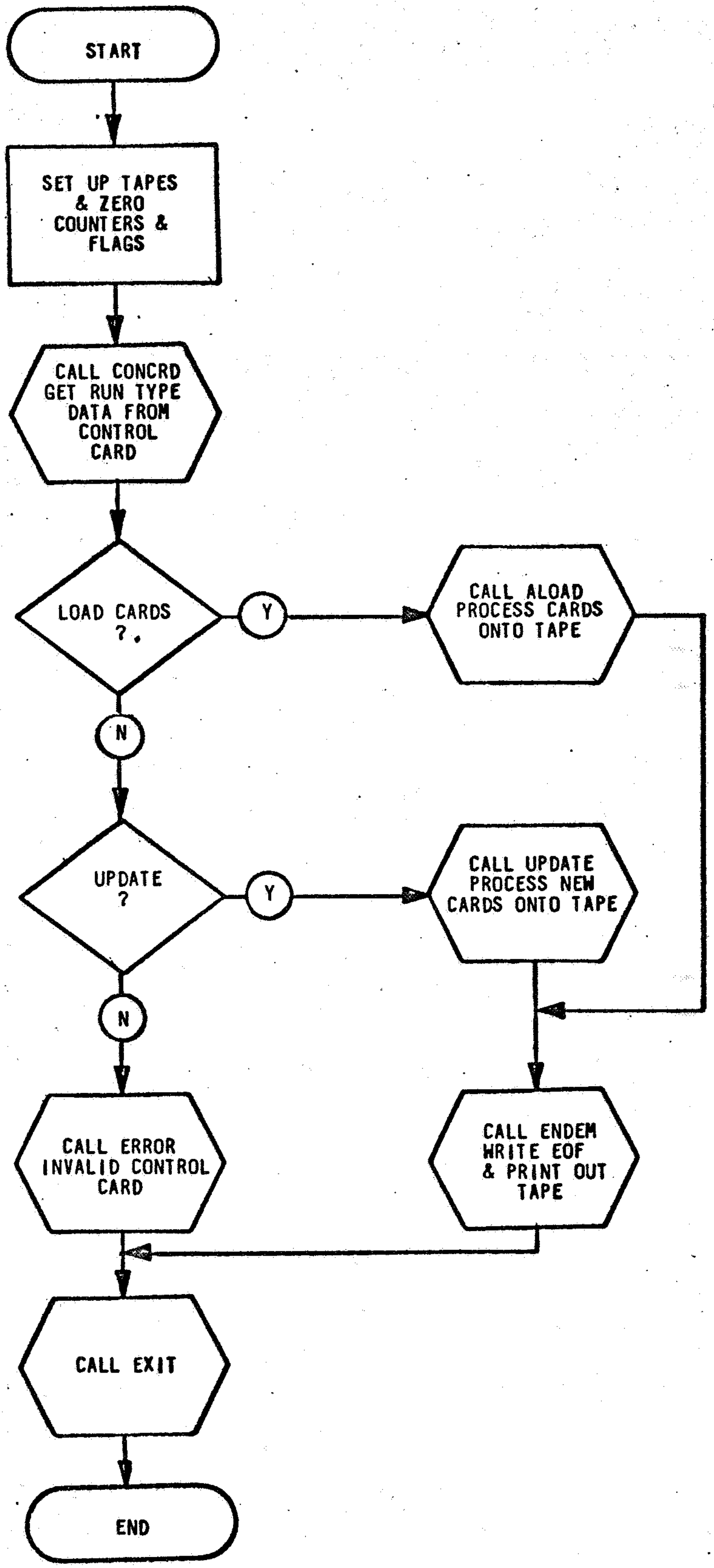

START
SET UP TAPES & ZERO COUNTERS & FLAGS
CALL CONCRD GET RUN TYPE DATA FROM CONTROL CARD
LOAD CARDS ?
Y
CALL ALOAD PROCESS CARDS ONTO TAPE
N
UPDATE ?
Y
CALL UPDATE PROCESS NEW CARDS ONTO TAPE
N
CALL ERROR INVALID CONTROL CARD
CALL ENDEM WRITE EOF & PRINT OUT TAPE
CALL EXIT
END

Figure A-1. SUBROUTINE ASETUP (1 of 1)

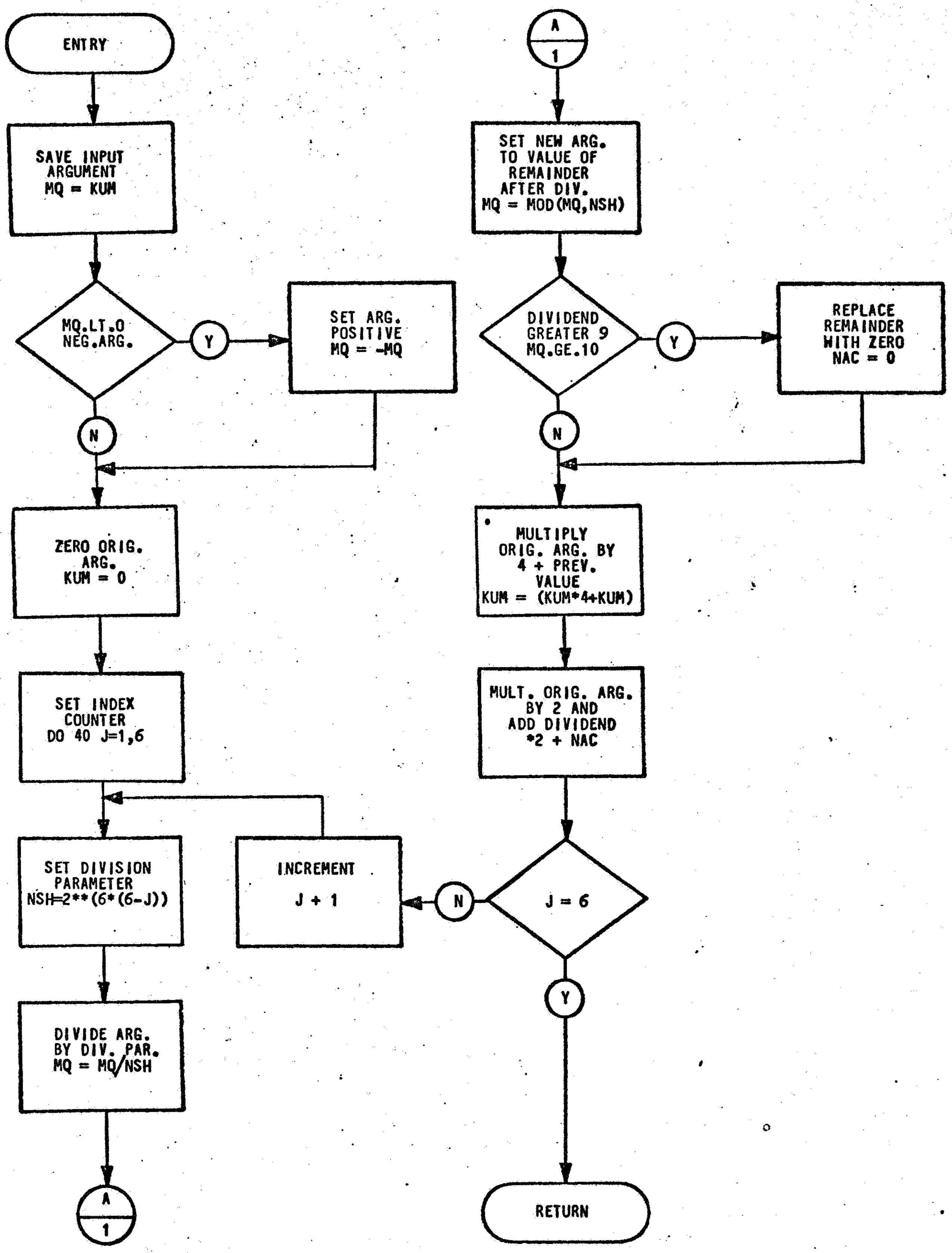

ENTRY
SAVE INPUT ARGUMENT MQ = KUM
MQ.LT.0 NEG.ARG.
Y
SET ARG. POSITIVE MQ = -MQ
N
ZERO ORIG. ARG. KUM = 0
SET INDEX COUNTER DO 40 J=1,6
SET DIVISION PARAMETER NSH=2**(6*(6-J))
DIVIDE ARG. BY DIV. PAR. MQ = MQ/NSH
A 1
A 1
SET NEW ARG. TO VALUE OF REMAINDER AFTER DIV. MQ = MOD(MQ,NSH)
DIVIDEND GREATER 9 MQ.GE.10
Y
REPLACE REMAINDER WITH ZERO NAC = 0
N
MULTIPLY ORIG. ARG. BY 4 + PREV. VALUE KUM = (KUM*4+KUM)
MULT. ORIG. ARG. BY 2 AND ADD DIVIDEND *2 + NAC
J = 6
N
INCREMENT J + 1
Y
RETURN

Figure A-2. SUBROUTINE CD2BIN (1 of 1)

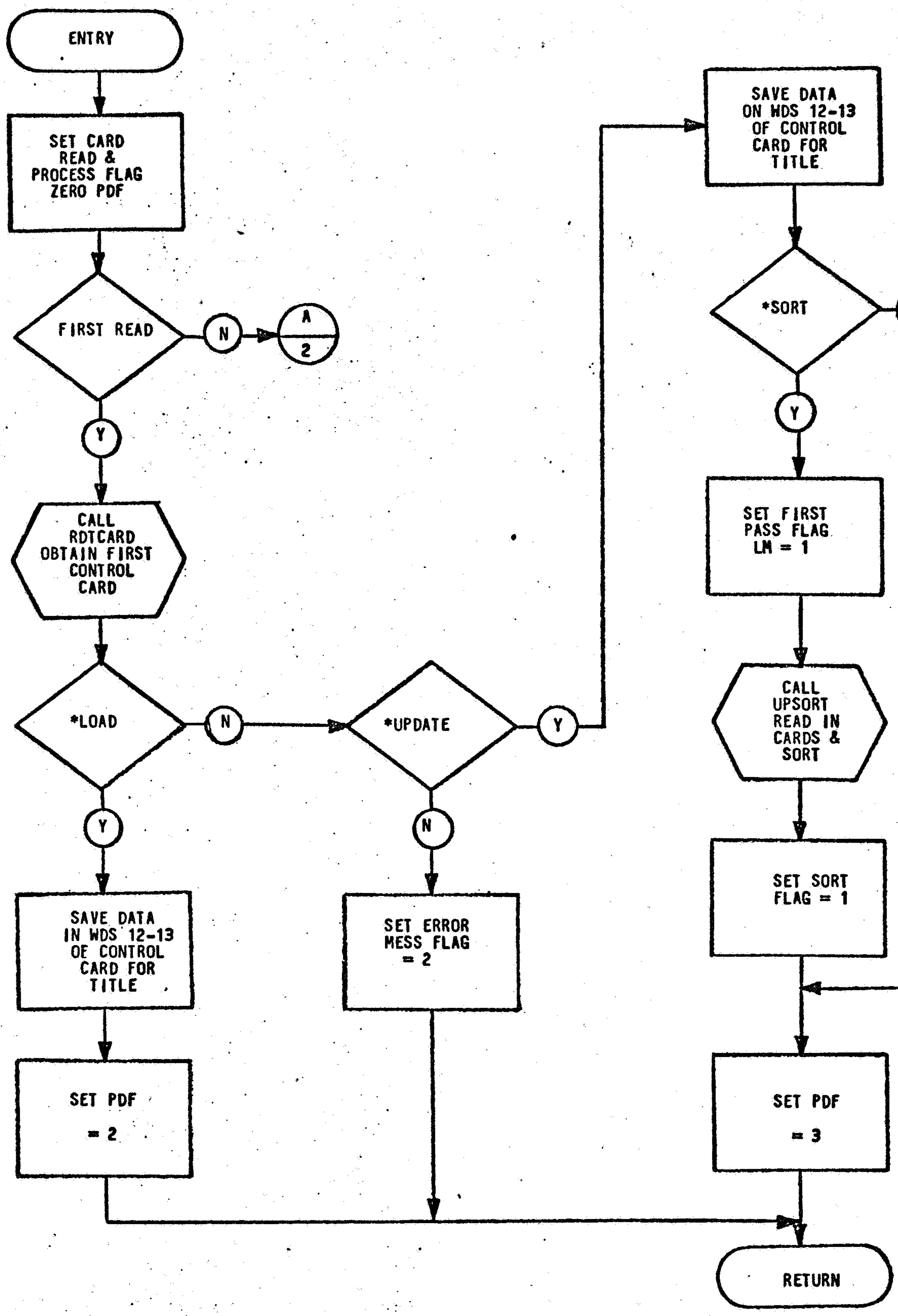

ENTRY
SET CARD READ & PROCESS FLAG ZERO PDF
FIRST READ
N
A
2
Y
CALL RDTCARD OBTAIN FIRST CONTROL CARD
*LOAD
N
*UPDATE
Y
N
Y
SAVE DATA IN WDS 12-13 OF CONTROL CARD FOR TITLE
SET PDF = 2
SET ERROR MESS FLAG = 2
SAVE DATA ON WDS 12-13 OF CONTROL CARD FOR TITLE
*SORT
N
Y
SET FIRST PASS FLAG LM = 1
CALL UPSORT READ IN CARDS & SORT
SET SORT FLAG = 1
SET PDF = 3
RETURN

Figure A-3. SUBROUTINE CONCRD (1 of 5)

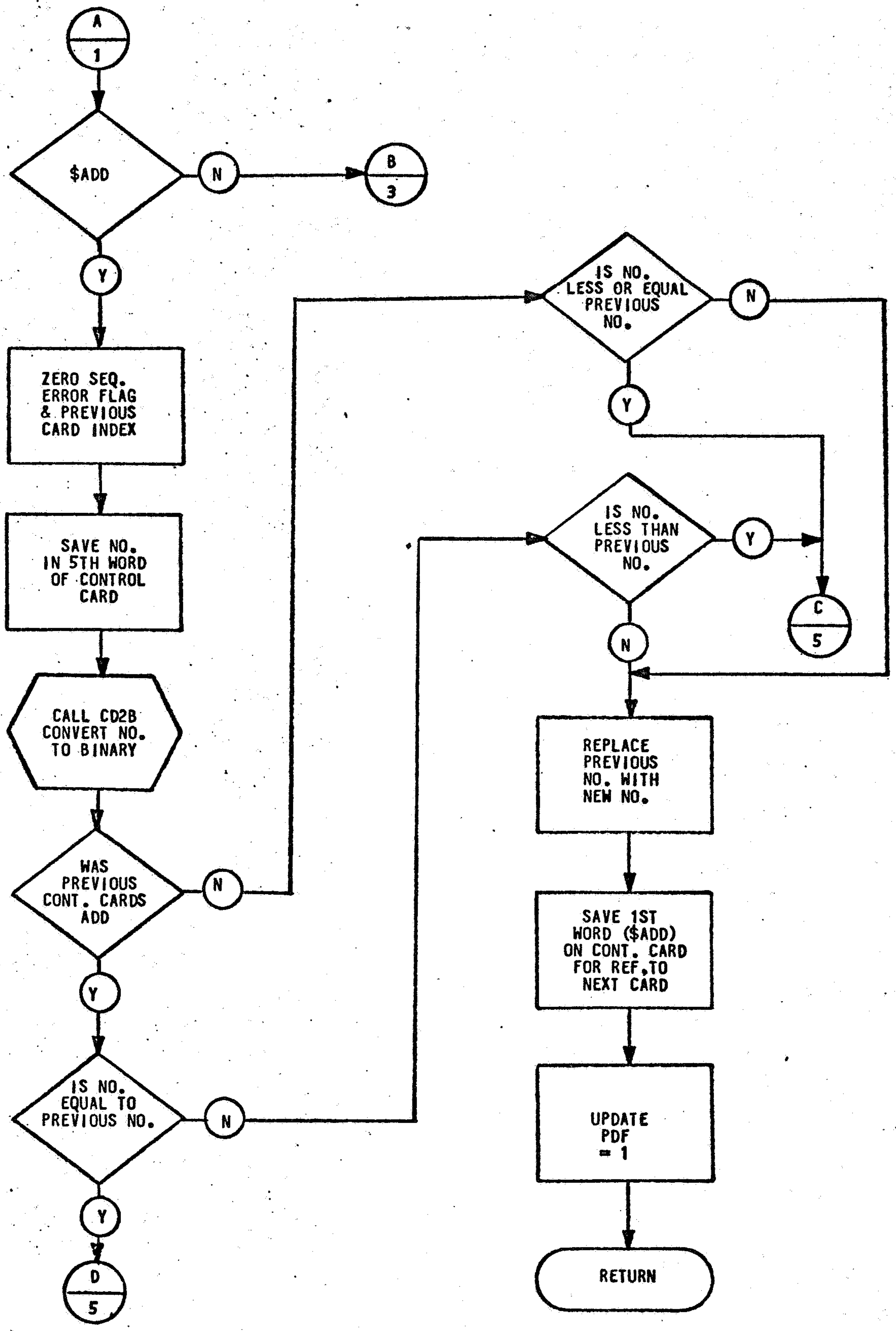
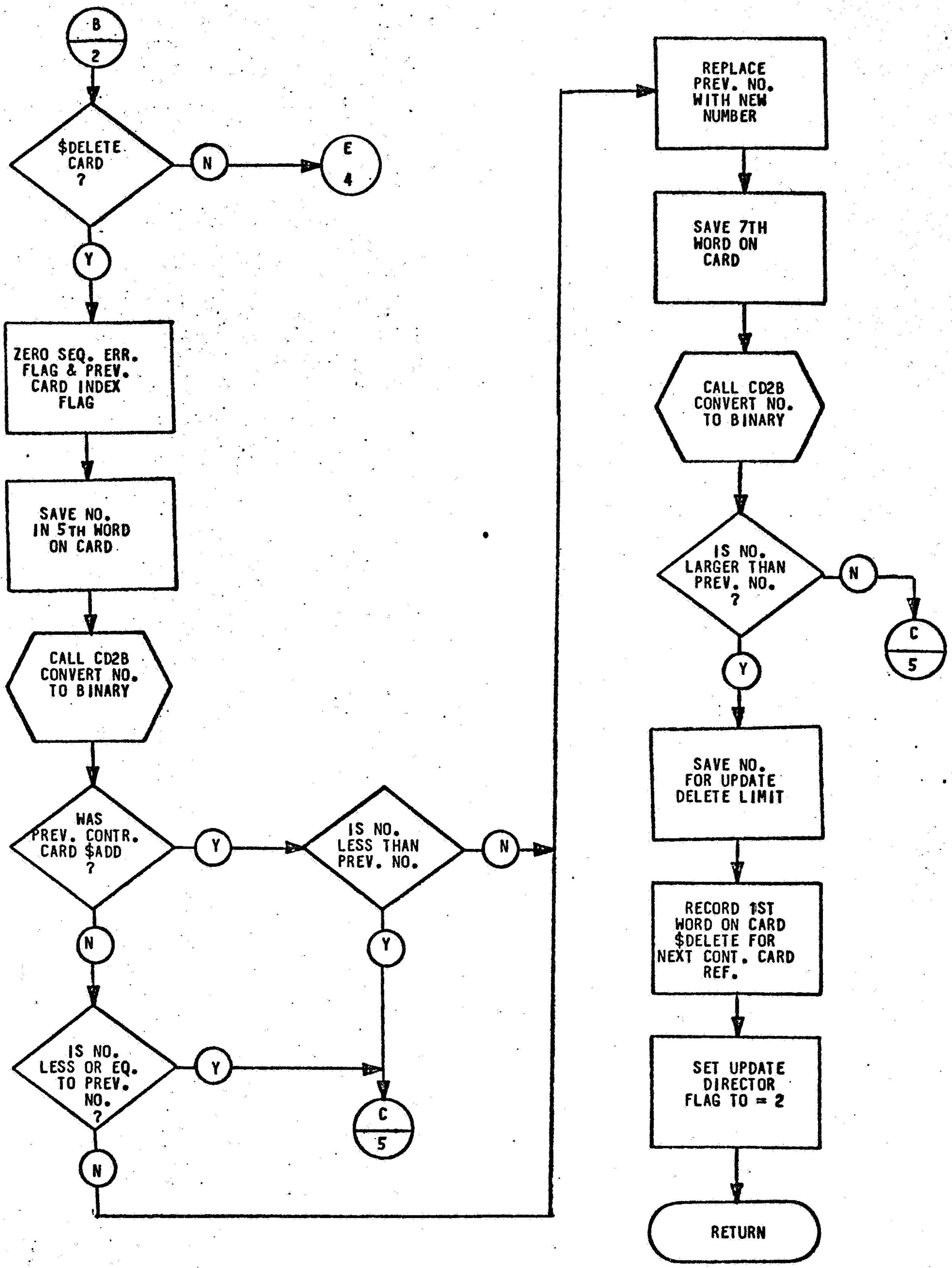

A
1
$ADD
N
B
3
Y
ZERO SEQ. ERROR FLAG & PREVIOUS CARD INDEX
SAVE NO. IN 5TH WORD OF CONTROL CARD
CALL CD2B CONVERT NO. TO BINARY
WAS PREVIOUS CONT. CARDS ADD
N
Y
IS NO. EQUAL TO PREVIOUS NO.
N
Y
D
5
IS NO. LESS OR EQUAL PREVIOUS NO.
N
Y
IS NO. LESS THAN PREVIOUS NO.
Y
C
5
N
REPLACE PREVIOUS NO. WITH NEW NO.
SAVE 1ST WORD ($ADD) ON CONT. CARD FOR REF. TO NEXT CARD
UPDATE PDF = 1
RETURN

Figure A-3. SUBROUTINE CONCRD (2 of 5)



B
2
$DELETE CARD ?
N
E
4
Y
ZERO SEQ. ERR. FLAG & PREV. CARD INDEX FLAG
SAVE NO. IN 5TH WORD ON CARD
CALL CD2B CONVERT NO. TO BINARY
WAS PREV. CONTR. CARD $ADD ?
Y
IS NO. LESS THAN PREV. NO.
N
Y
N
IS NO. LESS OR EQ. TO PREV. NO. ?
Y
C
5
N
REPLACE PREV. NO. WITH NEW NUMBER
SAVE 7TH WORD ON CARD
CALL CD2B CONVERT NO. TO BINARY
IS NO. LARGER THAN PREV. NO. ?
N
C
5
Y
SAVE NO. FOR UPDATE DELETE LIMIT
RECORD 1ST WORD ON CARD $DELETE FOR NEXT CONT. CARD REF.
SET UPDATE DIRECTOR FLAG TO = 2
RETURN

Figure A-3. SUBROUTINE CONCRD (3 of 5)

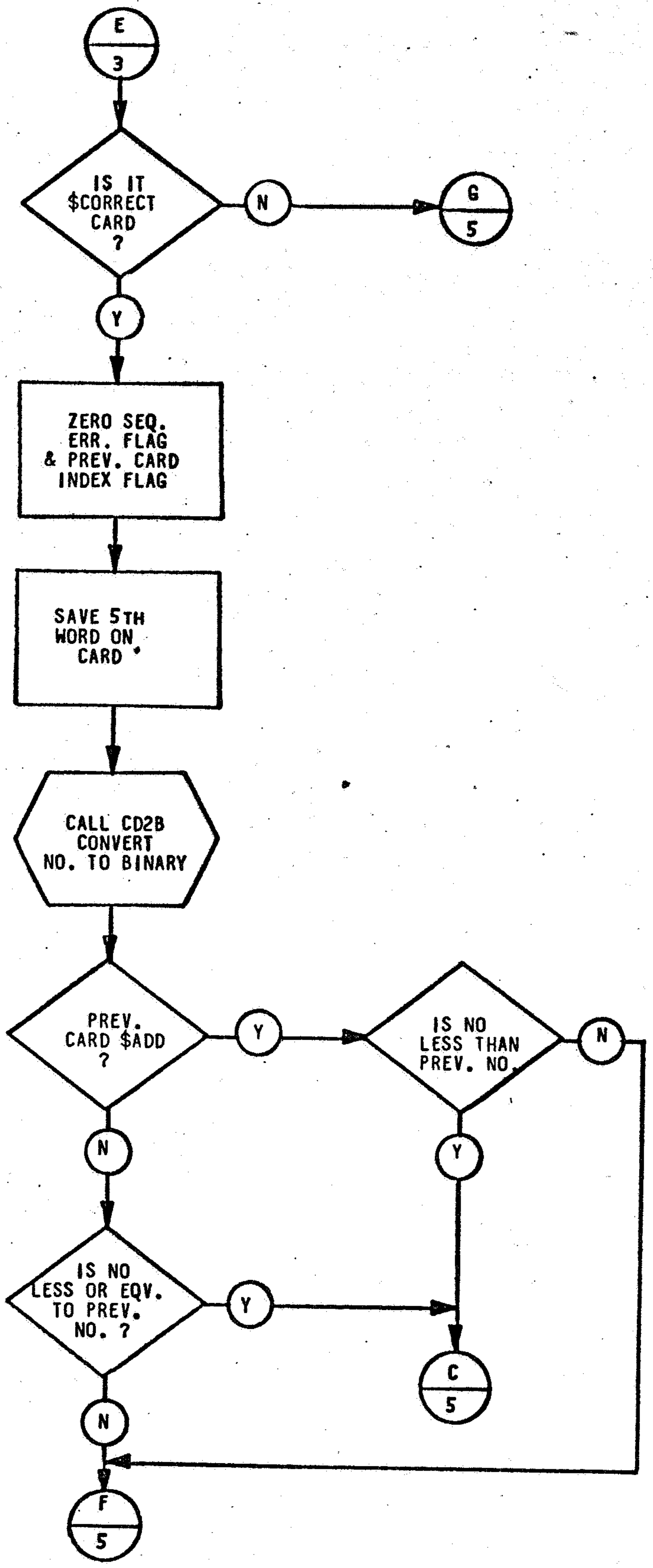

E
3
IS IT $CORRECT CARD ?
N
G
5
Y
ZERO SEQ. ERR. FLAG & PREV. CARD INDEX FLAG
SAVE 5TH WORD ON CARD
CALL CD2B CONVERT NO. TO BINARY
PREV. CARD $ADD ?
Y
IS NO LESS THAN PREV. NO.
N
N
Y
IS NO LESS OR EQV. TO PREV. NO. ?
Y
C
5
N
F
5

Figure A-3. SUBROUTINE CONCRD (4 of 5)

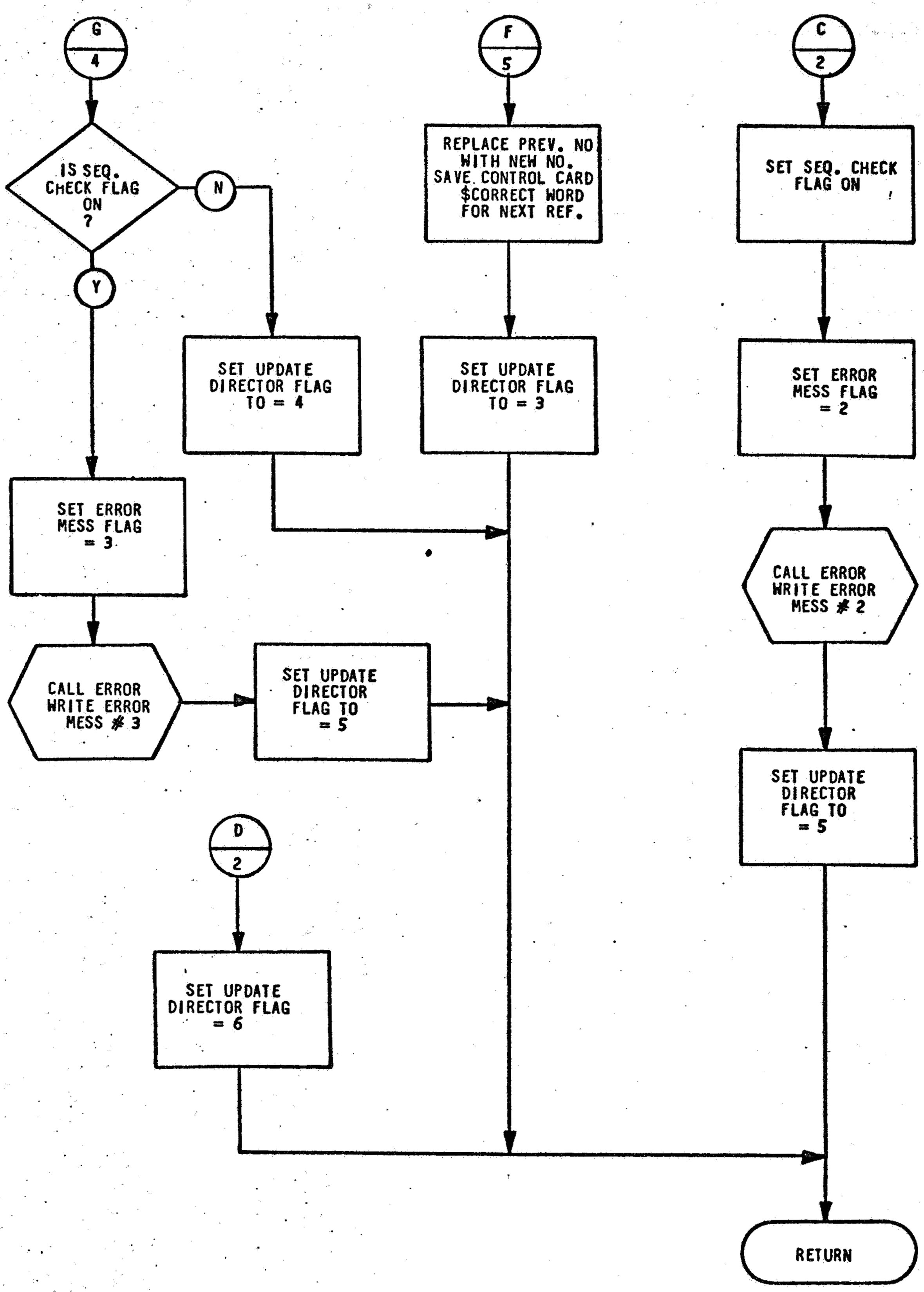

G
4
IS SEQ. CHECK FLAG ON ?
N
Y
SET UPDATE DIRECTOR FLAG TO = 4
SET ERROR MESS FLAG = 3
CALL ERROR WRITE ERROR MESS # 3
SET UPDATE DIRECTOR FLAG TO = 5
F
5
REPLACE PREV. NO WITH NEW NO. SAVE CONTROL CARD $CORRECT WORD FOR NEXT REF.
SET UPDATE DIRECTOR FLAG TO = 3
C
2
SET SEQ. CHECK FLAG ON
SET ERROR MESS FLAG = 2
CALL ERROR WRITE ERROR MESS # 2
SET UPDATE DIRECTOR FLAG TO = 5
D
2
SET UPDATE DIRECTOR FLAG = 6
RETURN

Figure A-3. SUBROUTINE CONCRD (5 of 5)

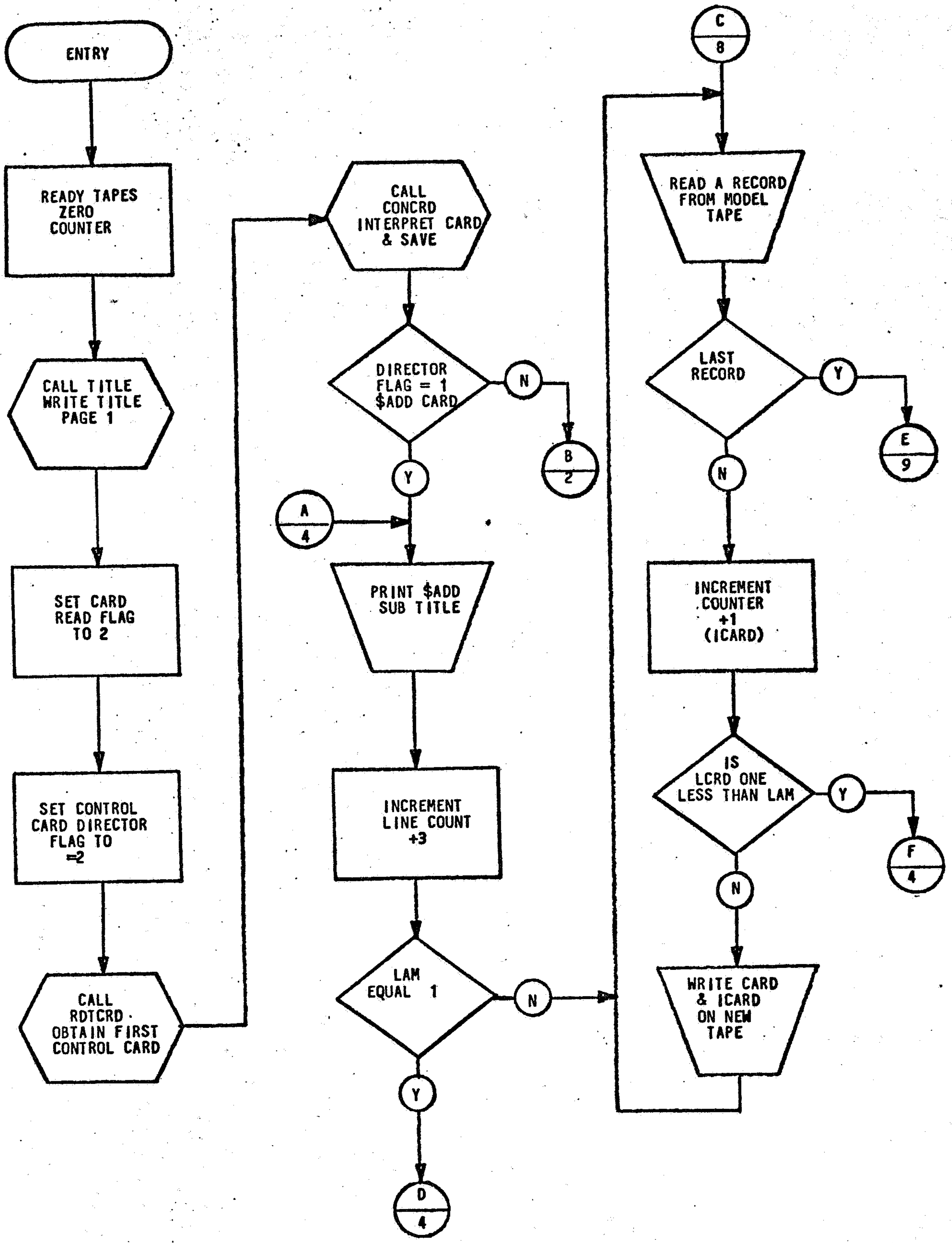

ENTRY
READY TAPES ZERO COUNTER
CALL TITLE WRITE TITLE PAGE 1
SET CARD READ FLAG TO 2
SET CONTROL CARD DIRECTOR FLAG TO =2
CALL RDTCRD OBTAIN FIRST CONTROL CARD
CALL CONCRD INTERPRET CARD & SAVE
DIRECTOR FLAG = 1 $ADD CARD
N
B 2
Y
A 4
PRINT $ADD SUB TITLE
INCREMENT LINE COUNT +3
LAM EQUAL 1
N
Y
D 4
C 8
READ A RECORD FROM MODEL TAPE
LAST RECORD
Y
E 9
N
INCREMENT COUNTER +1 (ICARD)
IS LCRD ONE LESS THAN LAM
Y
F 4
N
WRITE CARD & ICARD ON NEW TAPE

Figure A-4. SUBROUTINE CUPDAT (1 of 9)

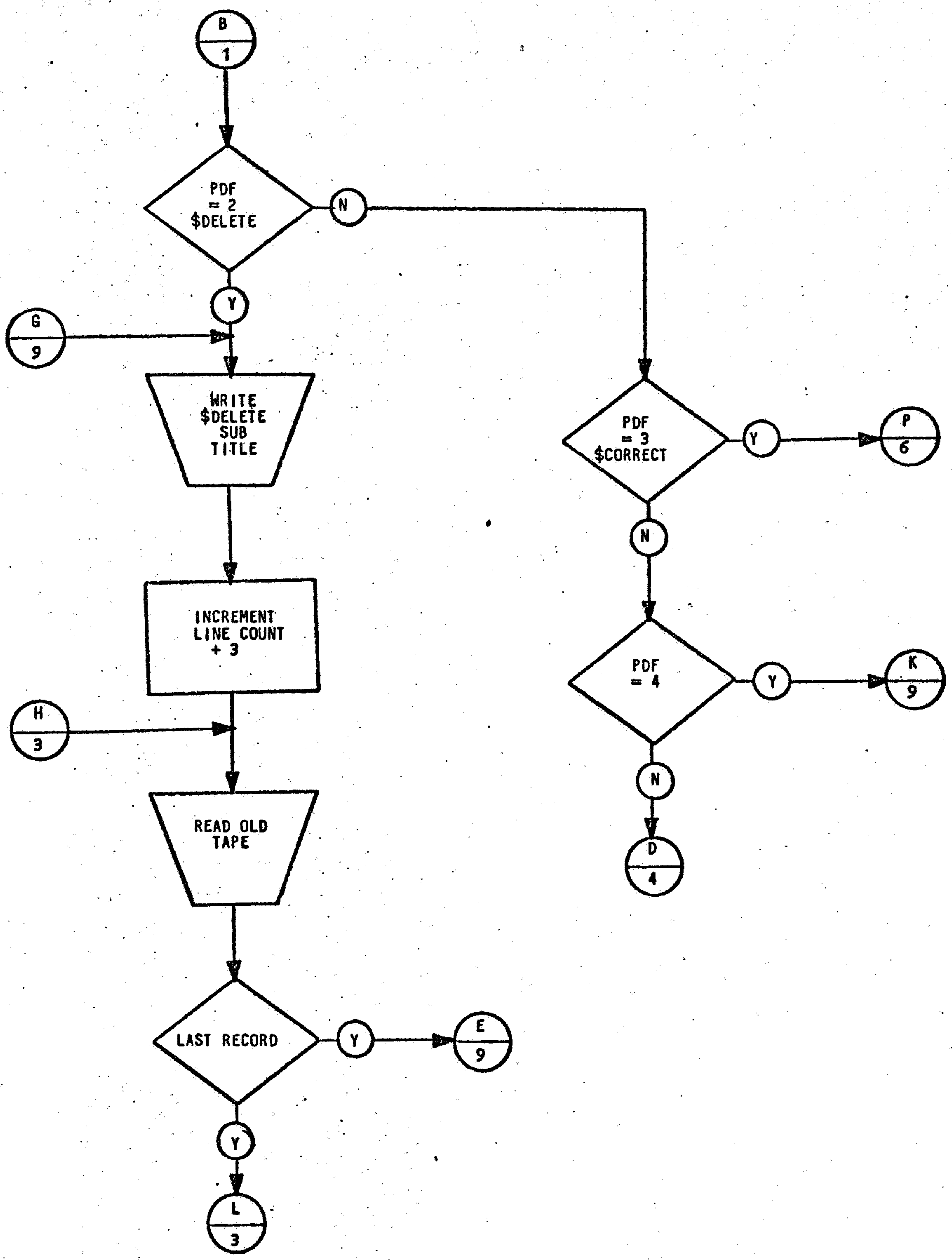


Figure A-4. SUBROUTINE CUPDAT (2 of 9)

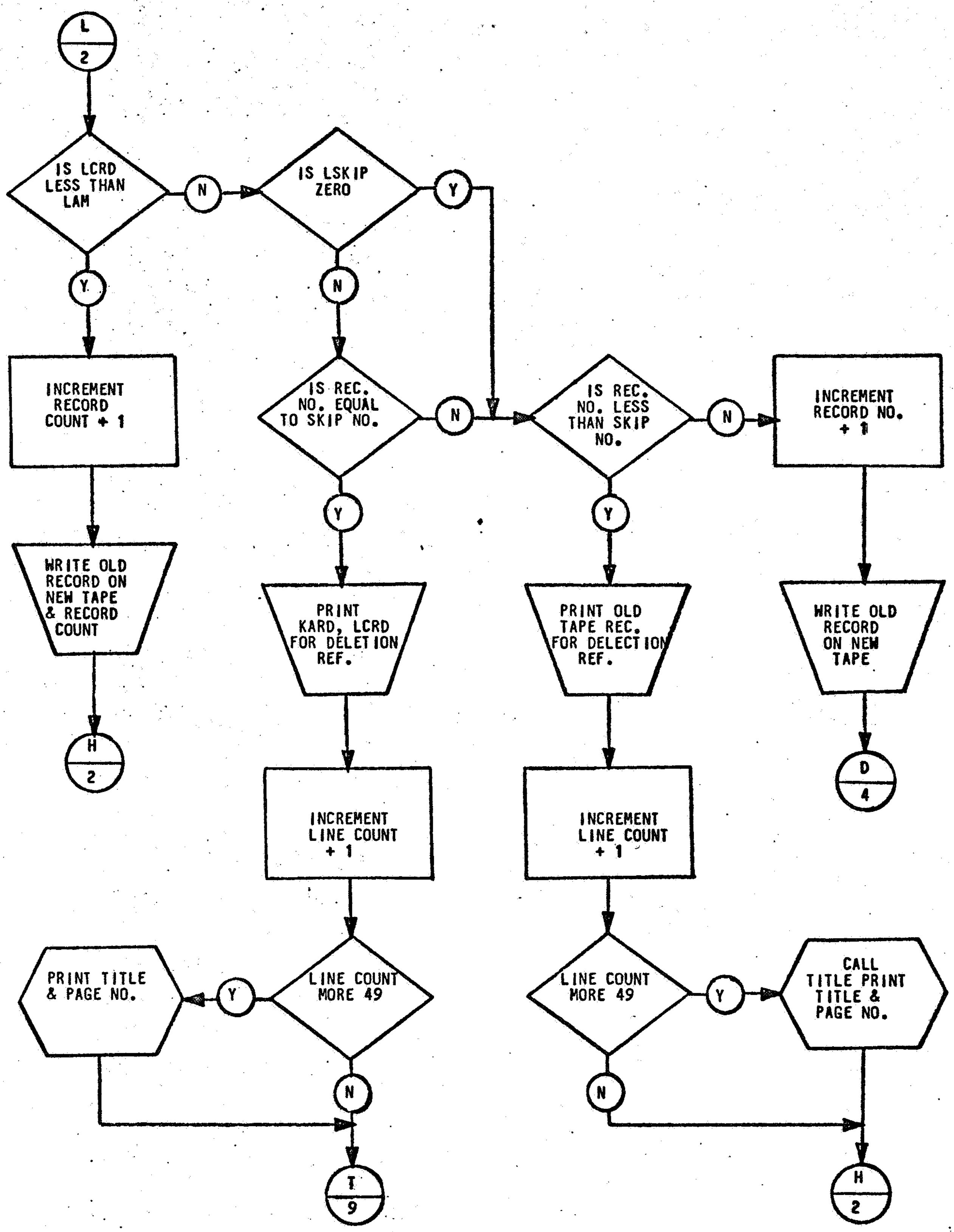

L 2
IS LCRD LESS THAN LAM
N
Y
IS LSKIP ZERO
Y
N
INCREMENT RECORD COUNT + 1
IS REC. NO. EQUAL TO SKIP NO.
N
Y
IS REC. NO. LESS THAN SKIP NO.
N
Y
INCREMENT RECORD NO. + 1
WRITE OLD RECORD ON NEW TAPE & RECORD COUNT
PRINT KARD, LCRD FOR DELETION REF.
PRINT OLD TAPE REC. FOR DELECTION REF.
WRITE OLD RECORD ON NEW TAPE
H 2
D 4
INCREMENT LINE COUNT + 1
INCREMENT LINE COUNT + 1
PRINT TITLE & PAGE NO.
Y
LINE COUNT MORE 49
N
LINE COUNT MORE 49
Y
CALL TITLE PRINT TITLE & PAGE NO.
N
T 9
H 2

Figure A-4. SUBROUTINE CUPDAT (3 of 9)

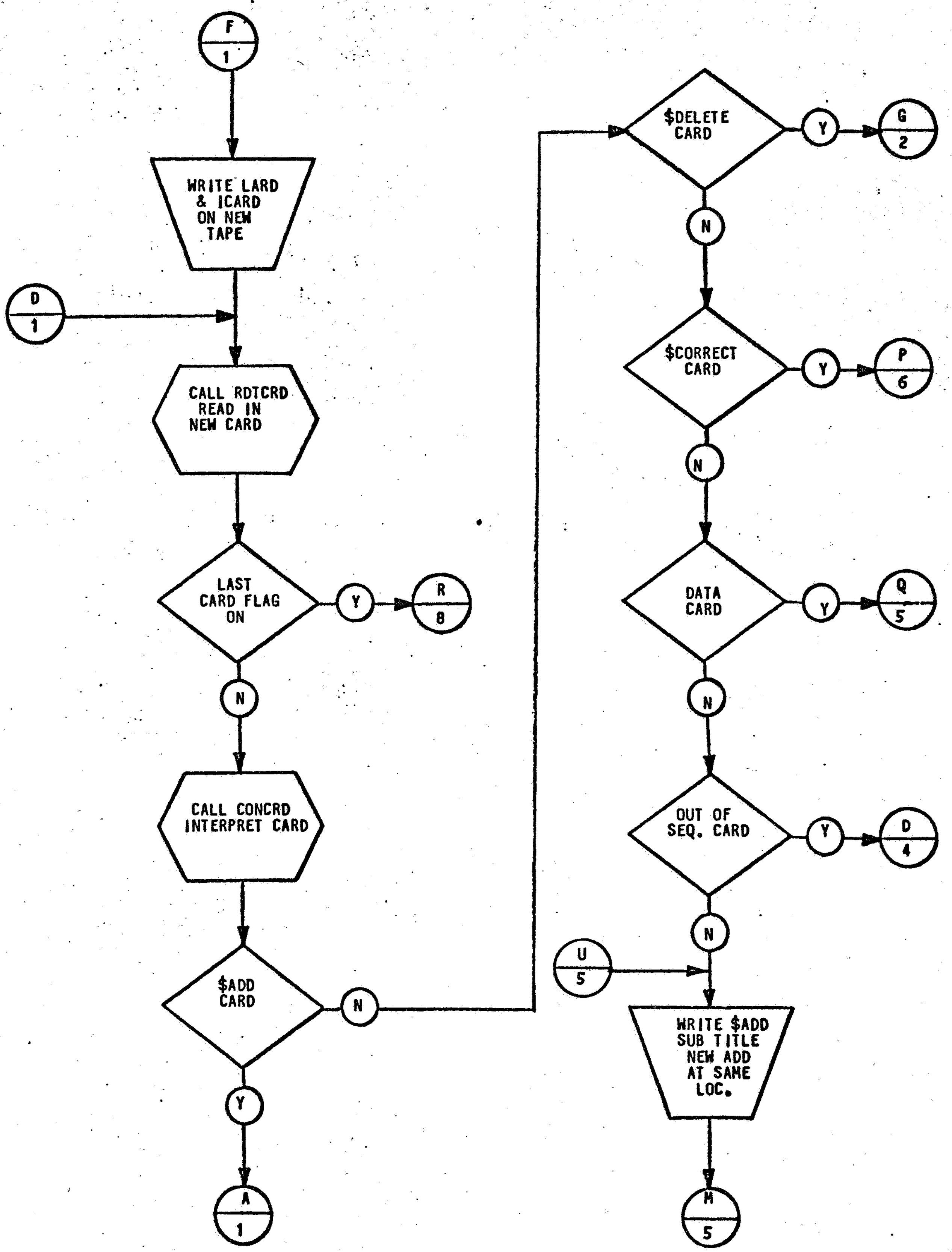

F 1
WRITE LARD & ICARD ON NEW TAPE
D 1
CALL RDTCRD READ IN NEW CARD
LAST CARD FLAG ON
Y
R 8
N
CALL CONCRD INTERPRET CARD
$ADD CARD
N
Y
A 1
$DELETE CARD
Y
G 2
N
$CORRECT CARD
Y
P 6
N
DATA CARD
Y
Q 5
N
OUT OF SEQ. CARD
Y
D 4
N
U 5
WRITE $ADD SUB TITLE NEW ADD AT SAME LOC.
M 5

Figure A-4. SUBROUTINE CUPDAT (4 of 9)

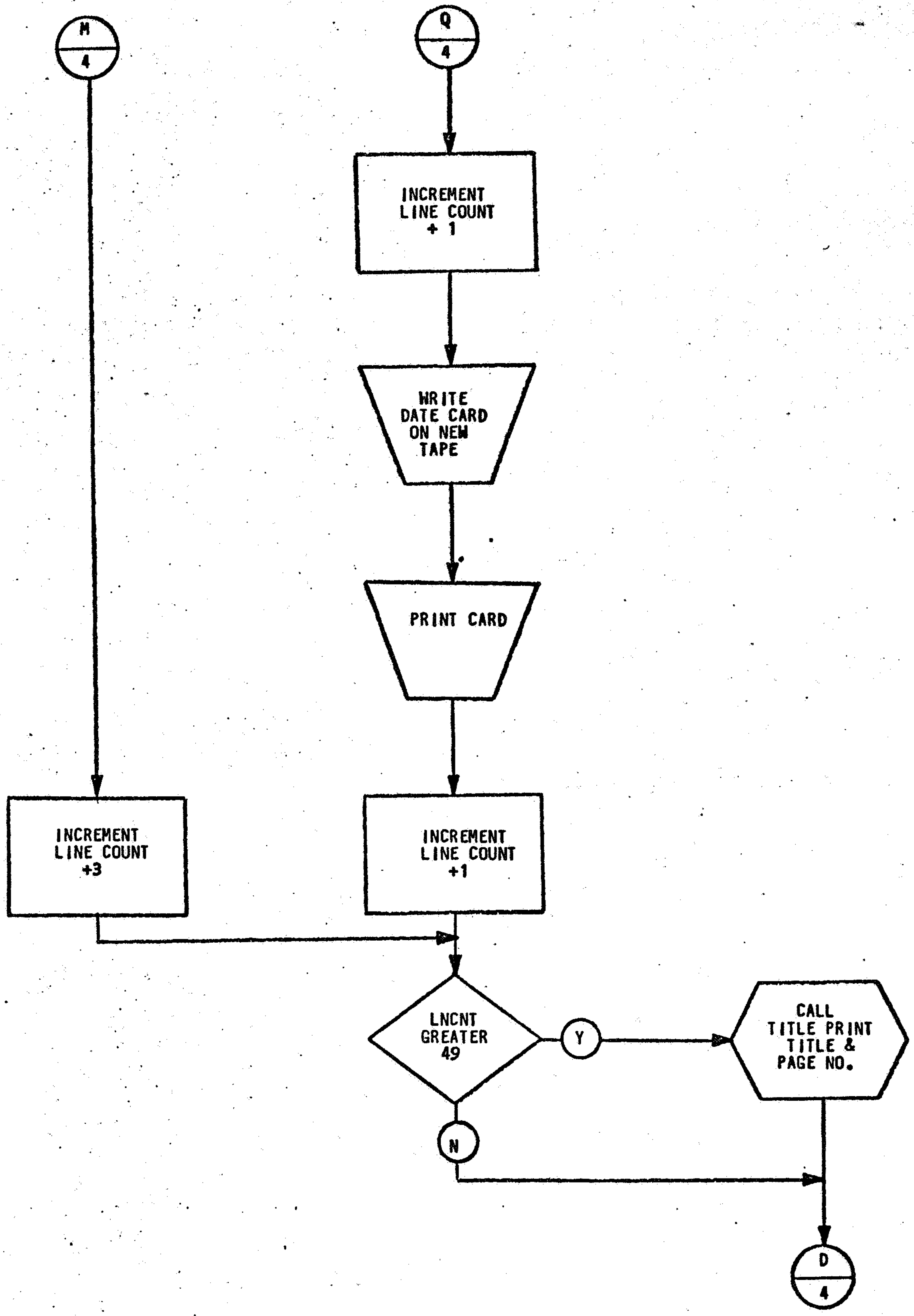


Figure A-4. SUBROUTINE CUPDAT (5 of 9)

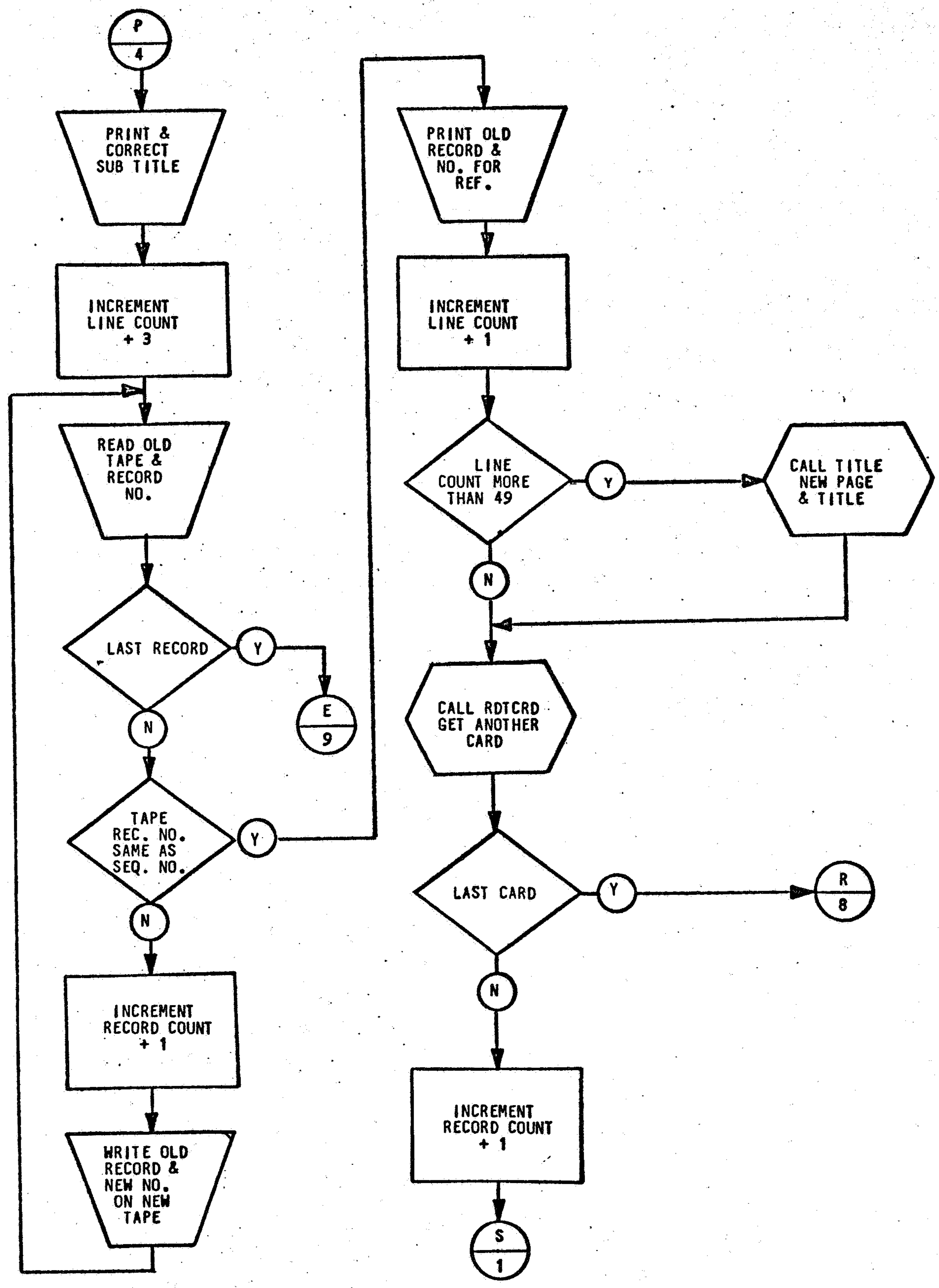

P
4
PRINT & CORRECT SUB TITLE
INCREMENT LINE COUNT + 3
READ OLD TAPE & RECORD NO.
LAST RECORD
Y
E
9
N
TAPE REC. NO. SAME AS SEQ. NO.
Y
N
INCREMENT RECORD COUNT + 1
WRITE OLD RECORD & NEW NO. ON NEW TAPE
PRINT OLD RECORD & NO. FOR REF.
INCREMENT LINE COUNT + 1
LINE COUNT MORE THAN 49
Y
CALL TITLE NEW PAGE & TITLE
N
CALL RDTCRD GET ANOTHER CARD
LAST CARD
Y
R
8
N
INCREMENT RECORD COUNT + 1
S
1

Figure A-4. SUBROUTINE CUPDAT (6 of 9)

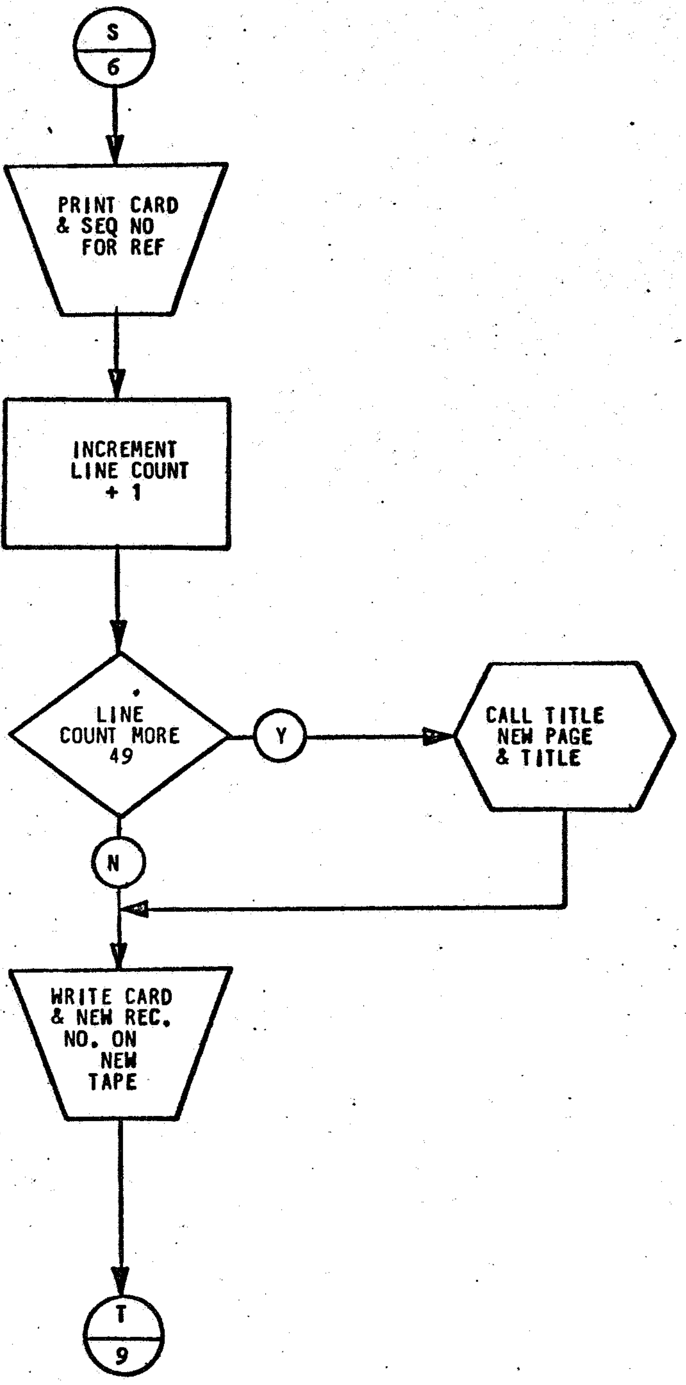

S
6
PRINT CARD & SEQ NO FOR REF
INCREMENT LINE COUNT + 1
LINE COUNT MORE 49
Y
CALL TITLE NEW PAGE & TITLE
N
WRITE CARD & NEW REC. NO. ON NEW TAPE
T
9

Figure A-4. SUBROUTINE CUPDAT (7 of 9)

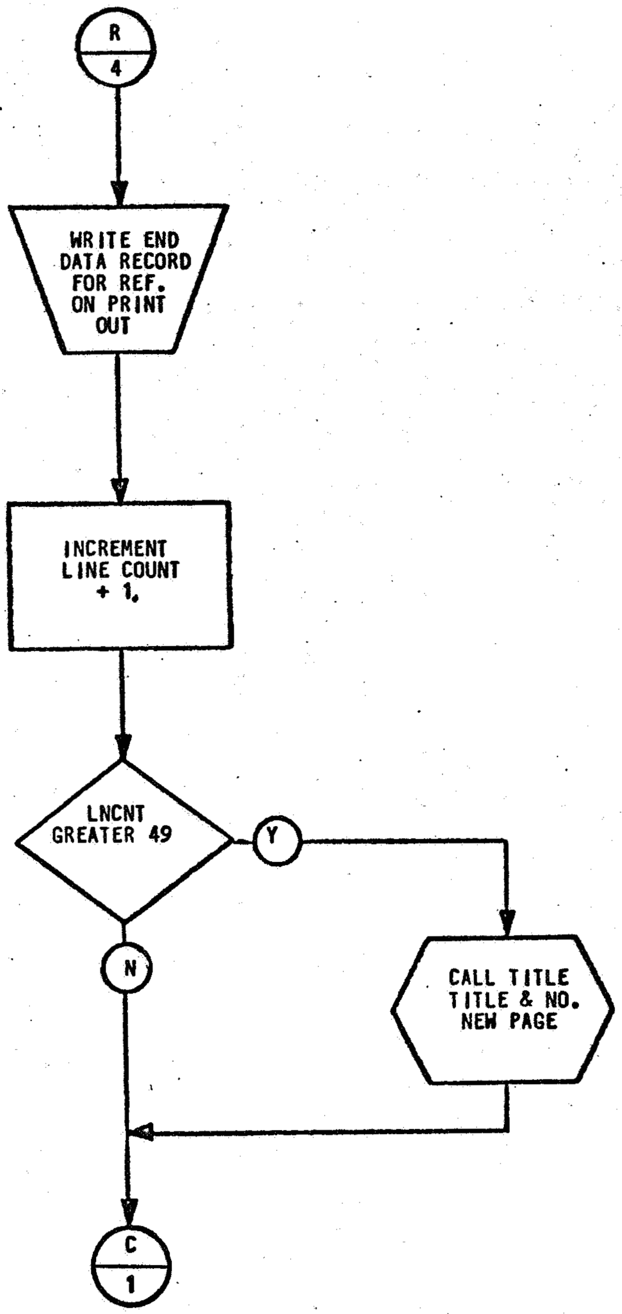

R
4
WRITE END DATA RECORD FOR REF. ON PRINT OUT
INCREMENT LINE COUNT + 1.
LNCNT GREATER 49
Y
CALL TITLE TITLE & NO. NEW PAGE
N
C
1

Figure A-4. SUBROUTINE CUPDAT (8 of 9)

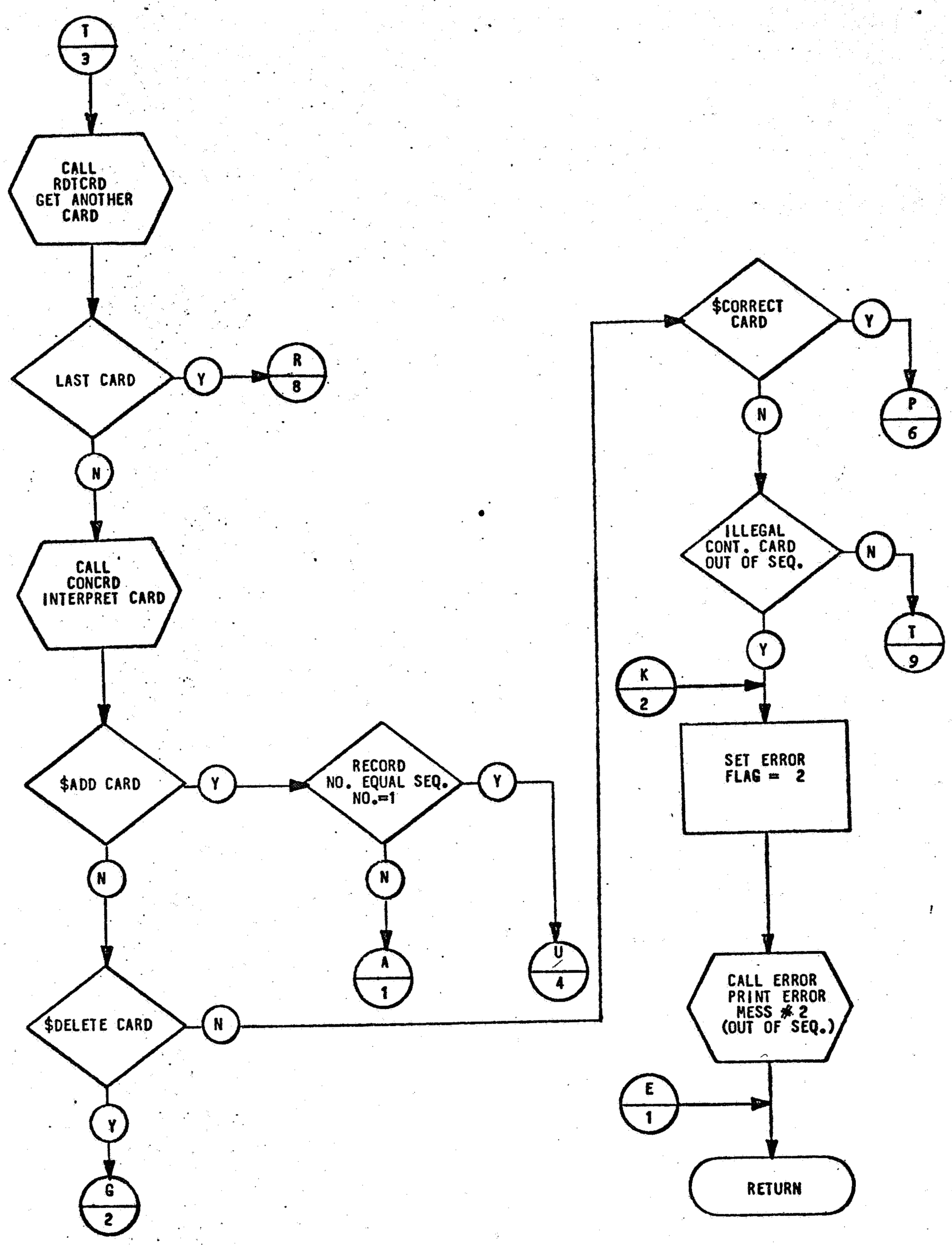

T 3
CALL RDTCRD GET ANOTHER CARD
LAST CARD
Y
R 8
N
CALL CONCRD INTERPRET CARD
$ADD CARD
Y
RECORD NO. EQUAL SEQ. NO.=1
Y
U 4
N
A 1
N
$DELETE CARD
Y
G 2
N
$CORRECT CARD
Y
P 6
N
ILLEGAL CONT. CARD OUT OF SEQ.
N
T 9
Y
K 2
SET ERROR FLAG = 2
CALL ERROR PRINT ERROR MESS # 2 (OUT OF SEQ.)
E 1
RETURN

Figure A-4. SUBROUTINE CUPDAT (9 of 9)

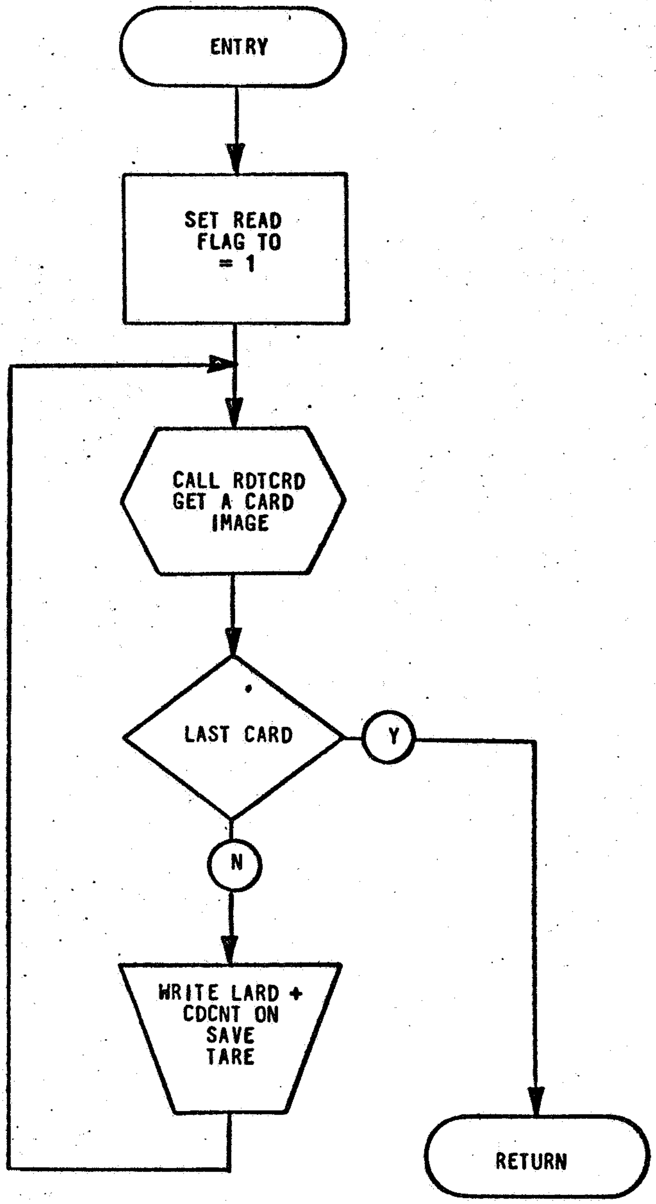


Figure A-5. SUBROUTINE GALOAD (1 of 1)

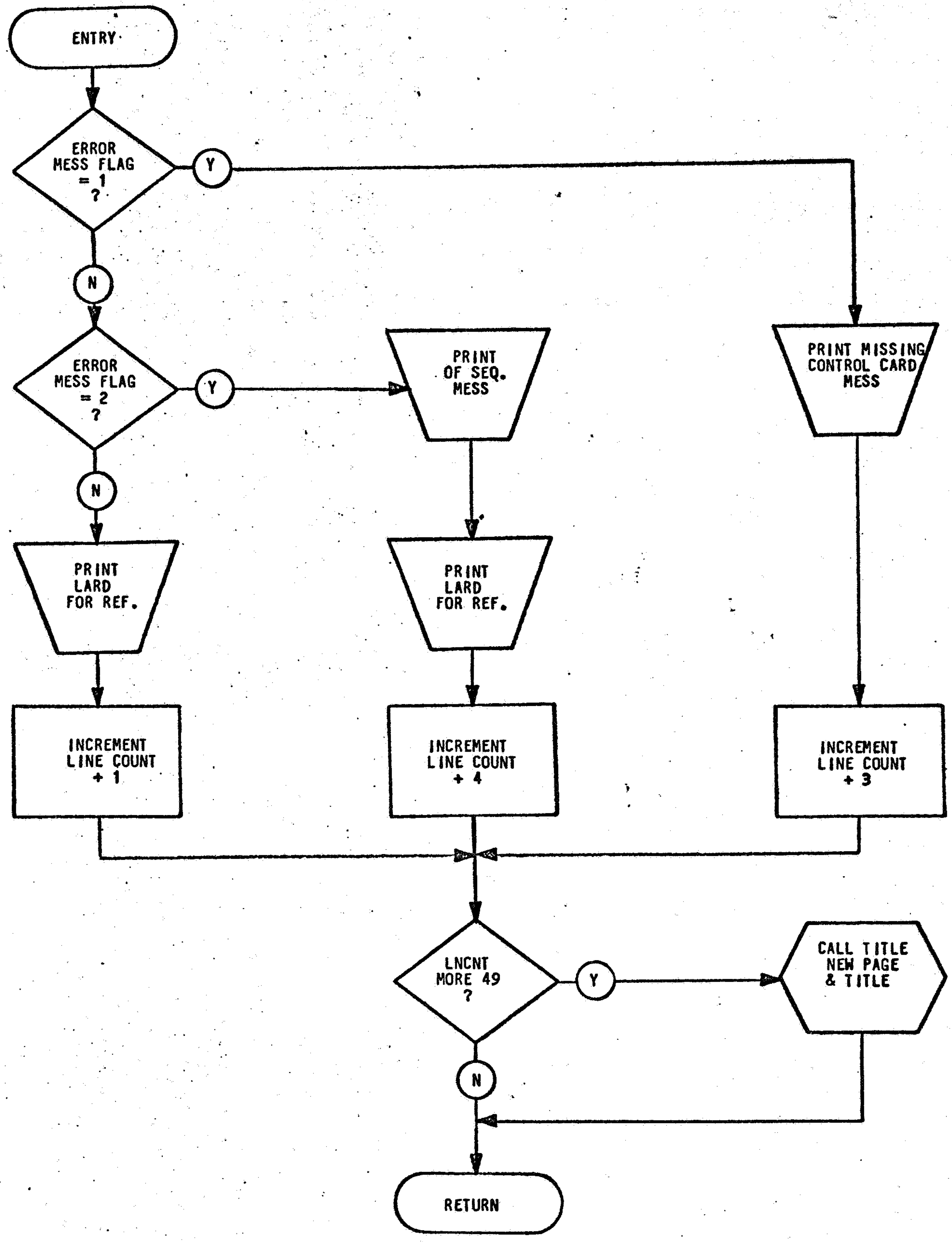


Figure A-6. SUBROUTINE HERROR (1 of 1)

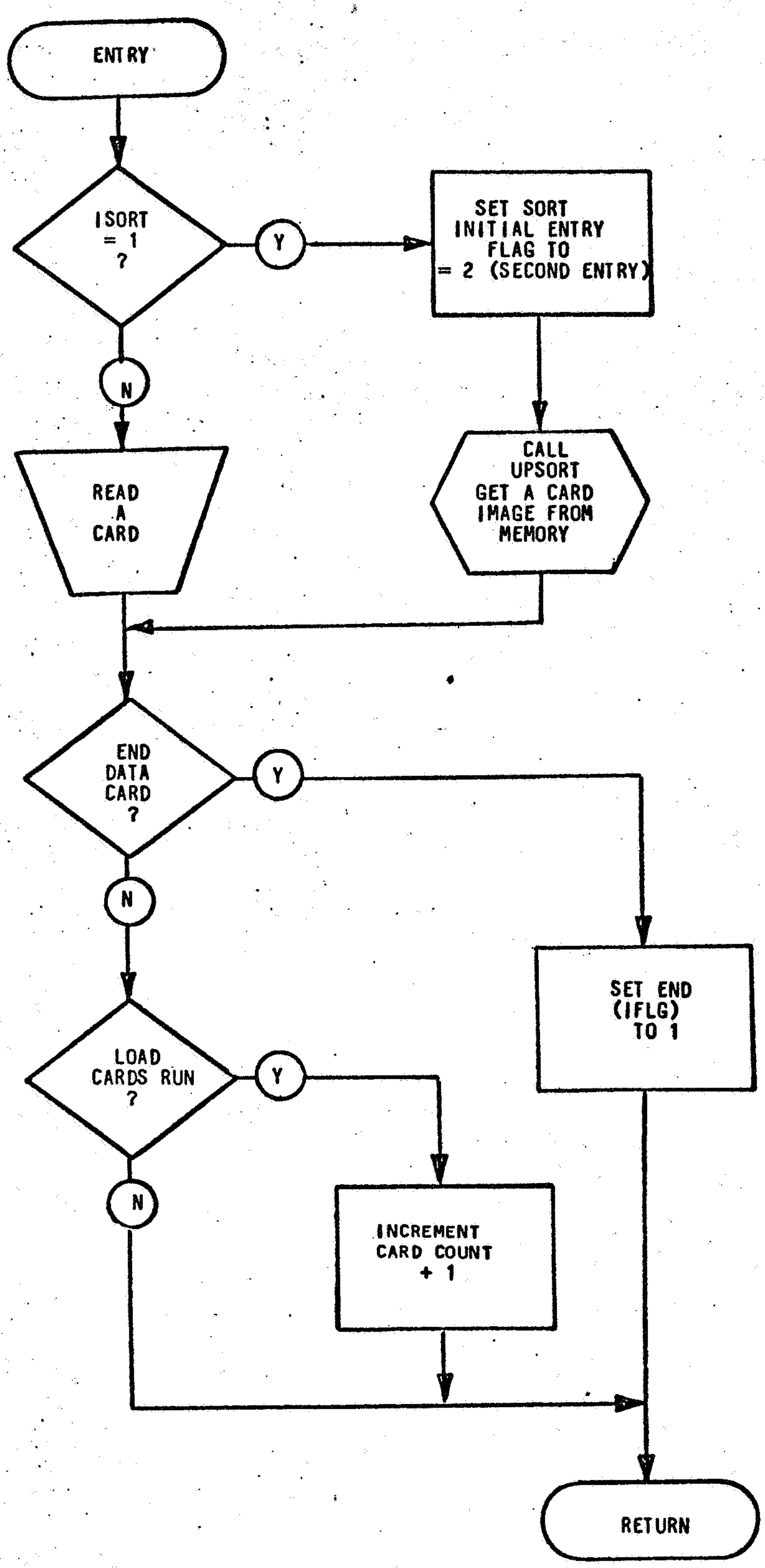


Figure A-7. SUBROUTINE RDTCRD (1 of 1)

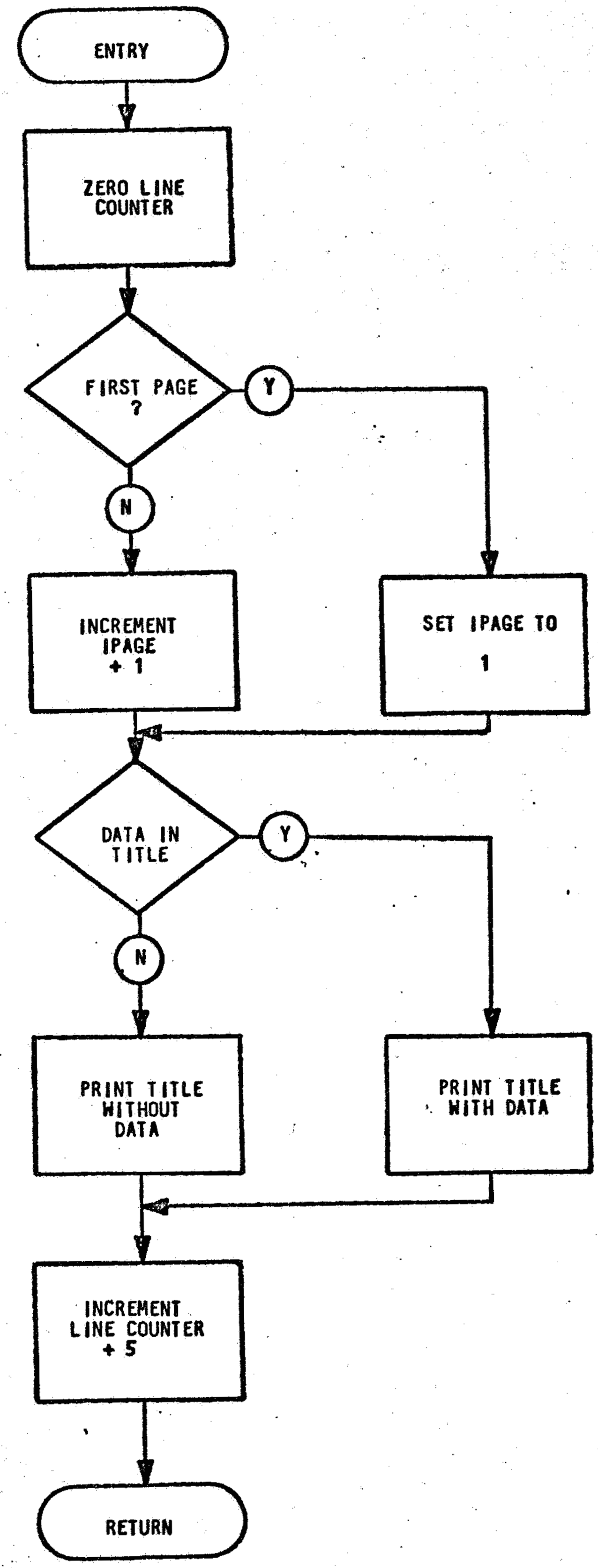


Figure A-8. SUBROUTINE TITLEX (1 of 1)

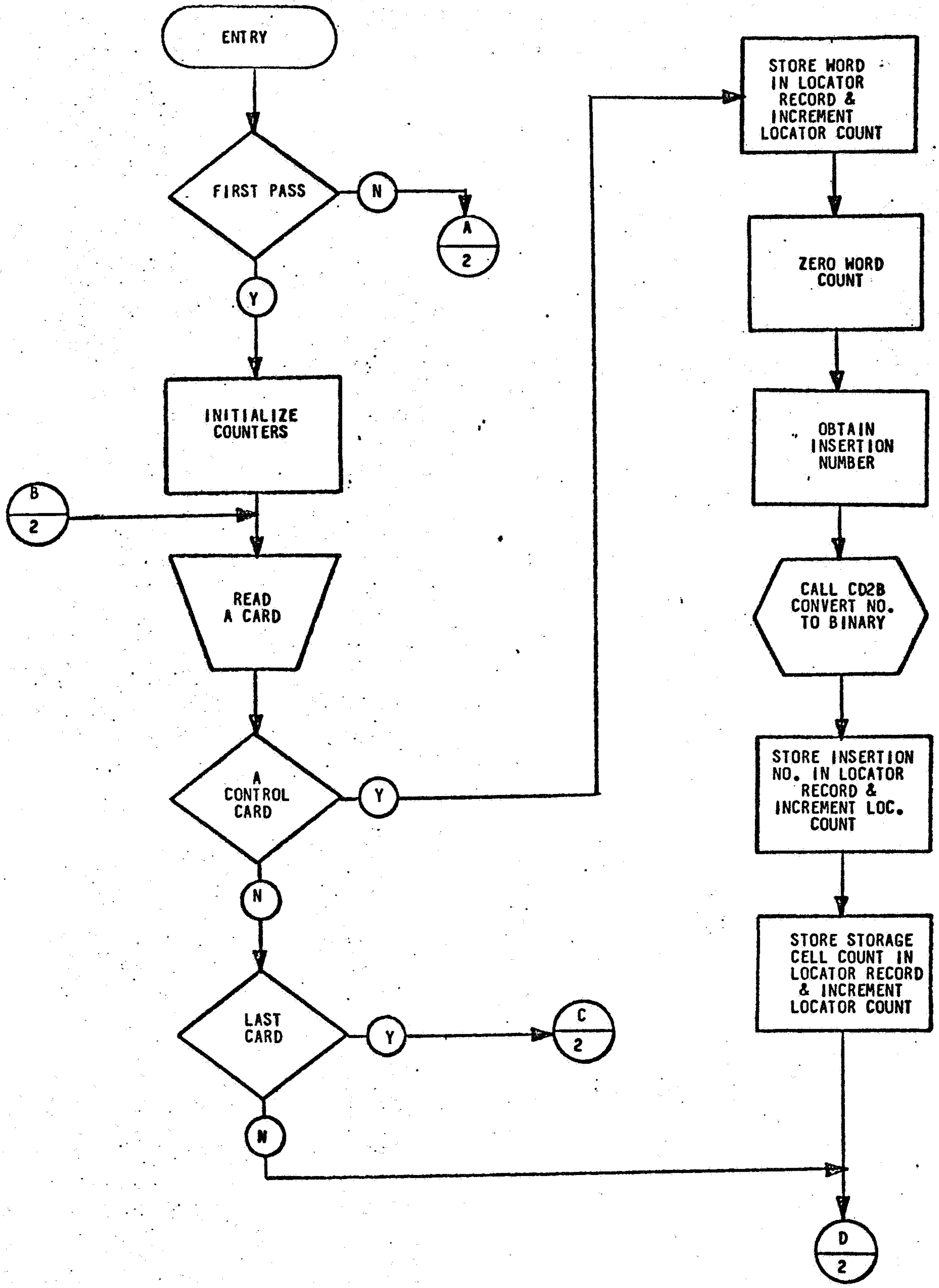

ENTRY
FIRST PASS
N
A
2
Y
INITIALIZE COUNTERS
B
2
READ A CARD
A CONTROL CARD
Y
N
LAST CARD
Y
C
2
N
STORE WORD IN LOCATOR RECORD & INCREMENT LOCATOR COUNT
ZERO WORD COUNT
OBTAIN INSERTION NUMBER
CALL CD2B CONVERT NO. TO BINARY
STORE INSERTION NO. IN LOCATOR RECORD & INCREMENT LOC. COUNT
STORE STORAGE CELL COUNT IN LOCATOR RECORD & INCREMENT LOCATOR COUNT
D
2

Figure A-9. SUBROUTING UPSORT (1 of 2)

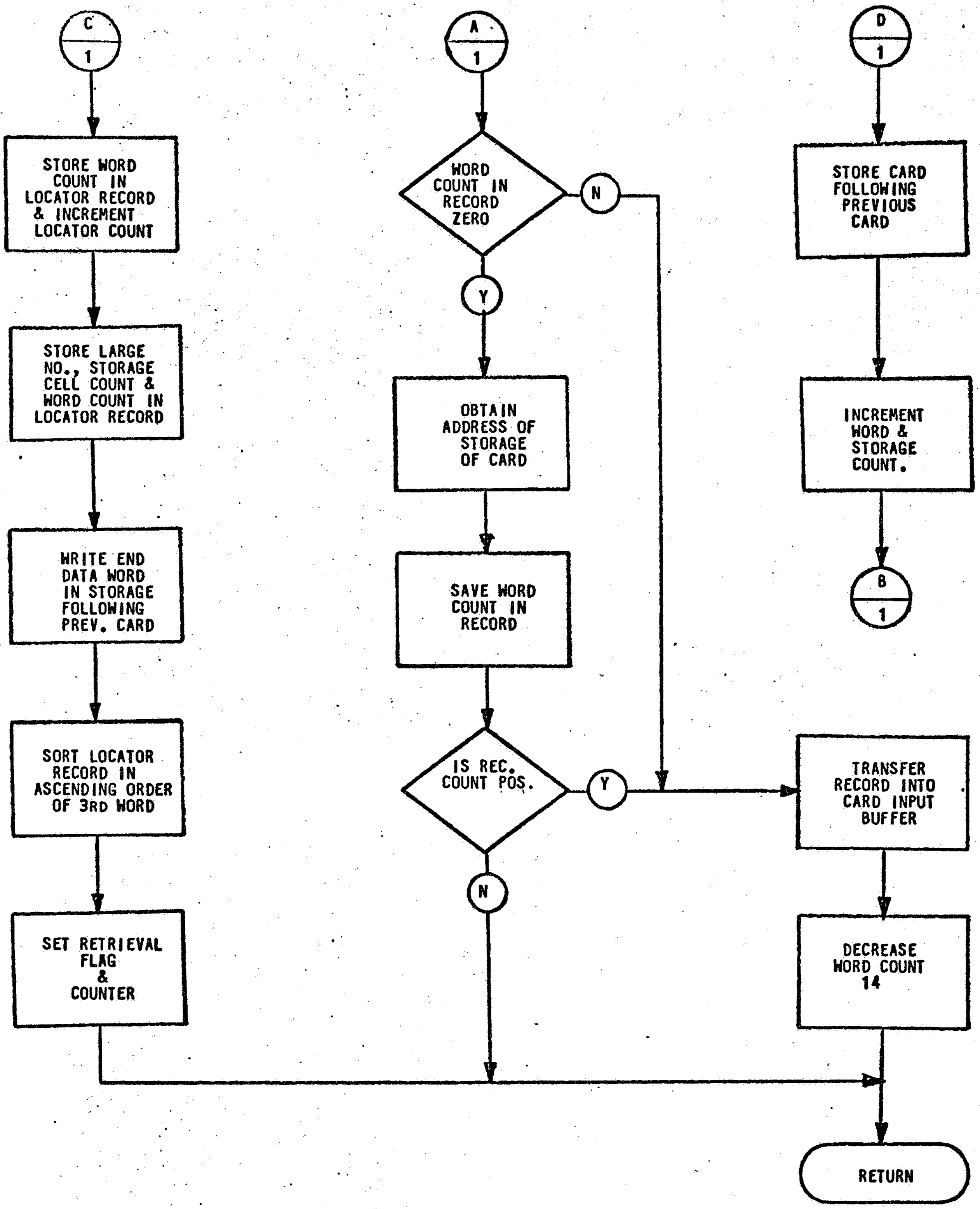

C 1
A 1
D 1
STORE WORD COUNT IN LOCATOR RECORD & INCREMENT LOCATOR COUNT
STORE LARGE NO., STORAGE CELL COUNT & WORD COUNT IN LOCATOR RECORD
WRITE END DATA WORD IN STORAGE FOLLOWING PREV. CARD
SORT LOCATOR RECORD IN ASCENDING ORDER OF 3RD WORD
SET RETRIEVAL FLAG & COUNTER
WORD COUNT IN RECORD ZERO
N
Y
OBTAIN ADDRESS OF STORAGE OF CARD
SAVE WORD COUNT IN RECORD
IS REC. COUNT POS.
Y
N
STORE CARD FOLLOWING PREVIOUS CARD
INCREMENT WORD & STORAGE COUNT.
B 1
TRANSFER RECORD INTO CARD INPUT BUFFER
DECREASE WORD COUNT 14
RETURN

Figure A-9. SUBROUTINE UPSORT (2 of 2)

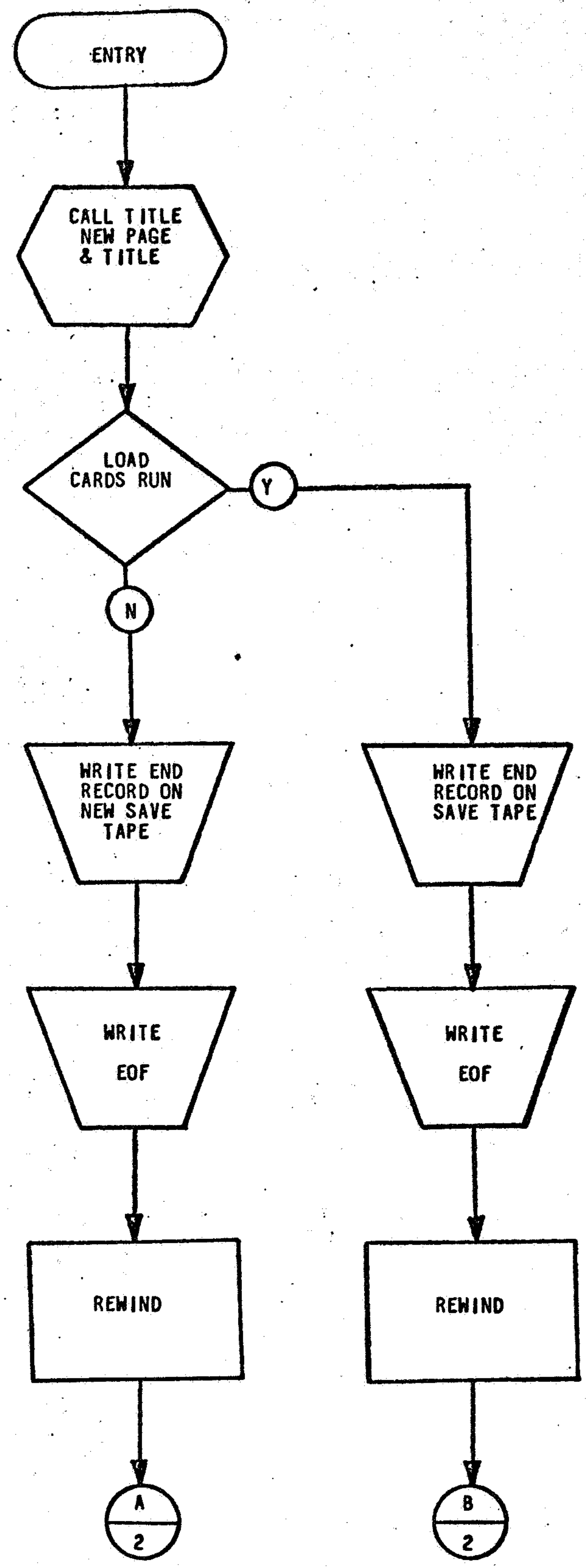

ENTRY
CALL TITLE NEW PAGE & TITLE
LOAD CARDS RUN
Y
N
WRITE END RECORD ON NEW SAVE TAPE
WRITE END RECORD ON SAVE TAPE
WRITE EOF
WRITE EOF
REWIND
REWIND
A 2
B 2

Figure A-10. SUBROUTINE ENDEM (1 of 2)

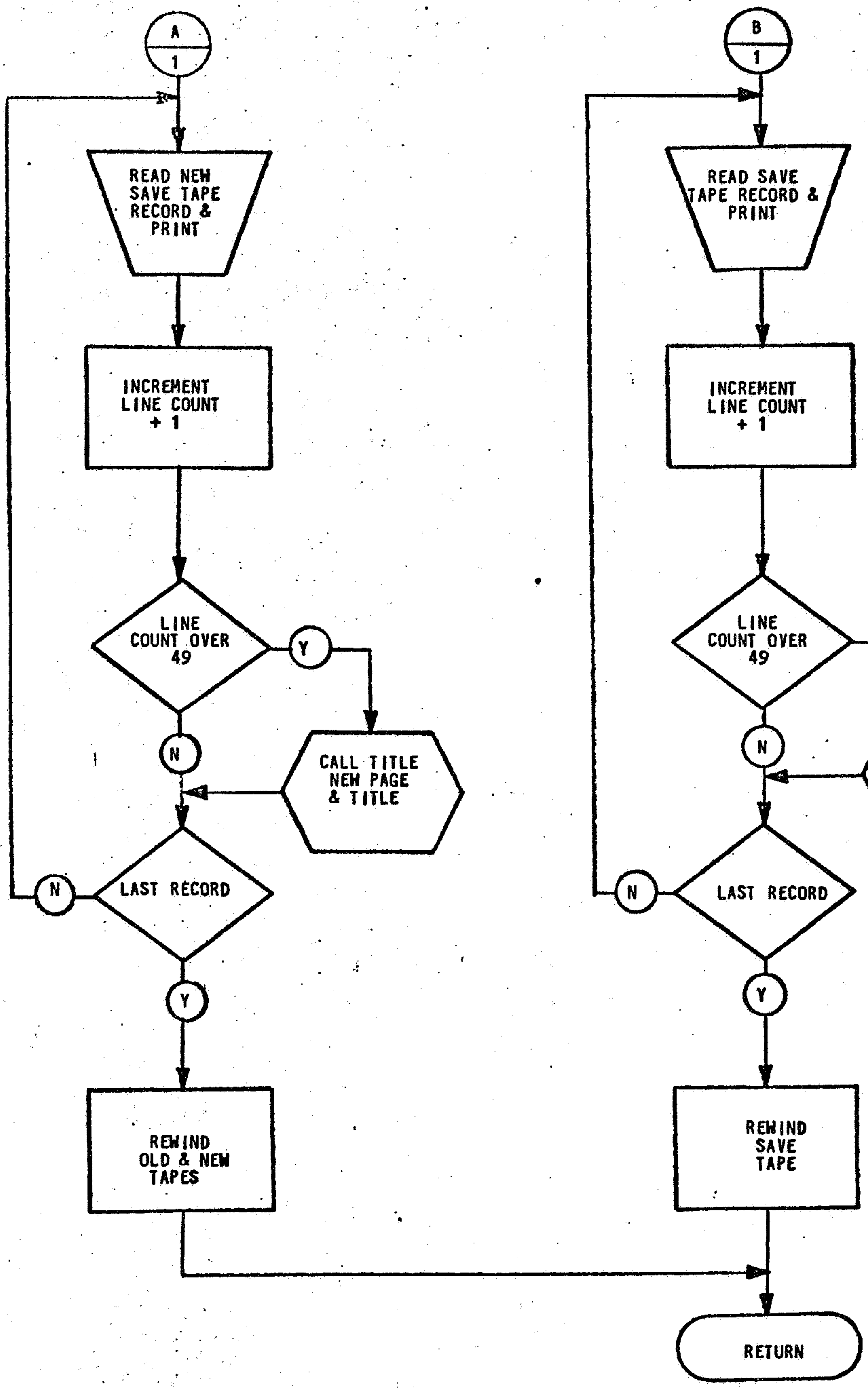


Figure A-10. SUBROUTINE ENDEM (2 of 2)

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLES

The following is an alphabetical listing of the terms used in the Update Program.

NAME	DESCRIPTION
CARDS	Tape I/O buffer.
CDCNT	Sequence number.
IADD	Word/Record count.
ICARD	Sequencing recount.
ICONT	SORT words/record count.
IFLG	End card enountered flag.
IKNTR	SORT storage cell counter.
ILOC	SORT retrieval word storage cell count.
INUM	Same as LAM/LSKIP after conversion.
IPCD	Temporary storage of UPDATE control card.
ISEQ	Out of sequence control card encountered flag.
ISNO	Update correction factor.
ISORT	SORT option selected flag.
ISTOR	SORT card internal storage.
ITEM	Temporary storage of retrieval word.
IWD	SORT retrieval word storage.
KARD	Tape I/O buffer storage.
LAM	Sequence number from UPDATE control card.
LARD	Card I/O buffer storage.
LCRD	Sequence number on tape record.
LNCNT	Printout line count.
LSKIP	Delete end sequence number from UPDATE control card.
LTAP/MTAP	Title words from initial control card.

2.0 DEFINITIONS

NAME	DESCRIPTION
CARDS	During reading and writing of new tape complete 15 word record on tape is transferred into CARDS and onto printout.
CDCNT	As each card is read, CDCNT is increased by one and is stored as 15th word on tape being created.
IADD	Record count of words during SORT card image retrieval. Value is found in word 3 of each set of retrieval records and is reduced by 14 as a 14 word card record is transferred into location LARD. A zero value in IADD signals that a new retrieval record must be obtained to find location of next card image.
ICARD	Each time a tape record or card is read, ICARD is incremented by one. Stored as 15th word on newly generated tape.
ICONT	The count of the number of words stored in ISTORE that apply to a specific control card.
IFLG	In read card routine, if a card containing *END DATA is encountered flag is set to 1 to signal no more cards to be input. Flag is tested during update routine and when sensed to be on, program bypasses further read calls.
IKNTR	As cards are read into memory during a SORT run, IKNTR is incremented by one for each word entered. When a control card is being read the value or IKNTR for the first word of the control card is stored in the retrieval table IWD.
ILOC	Retrieval table word counter. SORT option.
INUM	Buffer cell for input of sequence number read from control card. Value in INUM is input to CD2B conversion routine and then transferred into LAM or LSKIP as applicable.

NAME	DESCRIPTION
IPCD	As each control card is located, the first word is saved in location IPCD. As the next control card is encountered, the first word is compared with IPCD to see if this newly encountered update instruction followed an ADD update instruction.
ISEQ	During control card interpretation if an out of sequence update number is encountered, ISEQ is set to one so that if data cards are related to the out of sequence control card they will be printed. Flag is turned off when the next control card is encountered.
ISNO	If previous control card was an ADD card, correction factor of one (ISNOF 1) is included in the comparison of sequence numbers to ensure the validity of the new update number.
ISORT	Flag is turned on if first control card read indicates sorting of the update cards is desired. Flag is sensed in subroutine RDTCRD to direct where card will be obtained. Flag off - from A2 (input tape), flag on - from memory as directed by retrieval table.
ISTOR	Internal storage of update data deck during sort option consists of $9,800)_{10}$ words.
ITEM	During sorting of retrieval table, three word record is stored in this location.
IWD	Retrieval table consists of $2100)_{10}$ words for storage of three word locator records. Word one is always zero. Sort option.
KARD	Same as LARD but for tape records.
LAM	Working value converted to binary of sequence update numbers on control cards.
LARD	Fourteen (14) word buffer for card image in memory during input/output and card interpretation.
LCRD	Fifteen (15) word on tape being read, is compared with input control card number 'LAM' to find place on tape to update.

NAME	DESCRIPTION
LNCNT	Counter for printout and paging.
LSKIP	Same as LAM but only used during delete.
LTAP/MTAP	Internal memory to retain information listed in CC 67-80 of first control card. Each time a title is printed for a new page, the information into these words is printed on the LH top of page.

SECTION 2

NAME/TIME CARD GENERATOR PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

2

NAME/TIME CARD GENERATOR PROGRAM

AUTHOR: A. R. Stone
Convair division of General Dynamics

PURPOSE: The NAME/TIME Card Generator Program compiles a list of variables from the equation cards of a system model, determines the required timing, and classification data for each variable in the list, and punches this information on cards to create the time card section for the model.

Manual preparation of the time parameter cards for a system model can involve considerable time and effort. Each variable that is included in the system model must have a time parameter card containing its name, its pickup and drop out times, and an activity code relating to its working state for Discrete Network Simulation (DNS). If the variable is to be considered as a candidate for Automatic Malfunction Analysis (AMA), the card must also contain the hardware classification code for the variable. Data for time parameter cards is normally written by the analyst on coding sheets and subsequently keypunched. Complete and thorough checks throughout this entire process are required to avoid errors. The data entries and format for these cards must be accurate if results from the subsequent DNS and AMA Programs are to be valid.

The technique utilizing the NAME/TIME Card Generator eliminates most of the manual effort required to accomplish these tasks.

STORAGE: The program occupies $42,171)_8$ consecutive locations in memory and consists of the following 6 subroutines:

1.	READ	Driver and mode selection
2.	NAMES	Generation of original tables and duplicate elimination
3.	NARRING	Comparison of LH and RH tables for duplicates, generates third table and packs LH and RH tables.
4.	NSORT	Sorts each table in alpha numeric order.
5.	NTYPE	Classification, time parameter and AMA code determination
6.	PRINZ	Printing of time cards and punch tape generation

TIMING: Approximately 500 time parameter cards will be processed for each minute of 7090/94 computer time.

USE: The Time and Name Card Punch Program can be used in three distinct modes. Tape requirements and control cards are listed for each mode.

Mode 1

1. First control card in data deck

Col. 1-6	must have *EQUAT
Col. 7-80	blank

2. Equation cards

3. Immediately following last equation card

Col. 1-6	must have *ENDbE
Col. 7-80	blank

This mode produces time cards in the format required for the DNS/Preprocessor Program.

Mode 2

1. First control card in data deck

Col. 1-6	must have *EQUAT
Col. 7-12	blank
Col. 13-18	must have DTCbbb
Col. 19-80	blank

2. Equation cards

3. Immediately following last equation card

Col. 1-6	must have *ENDbE
Col. 7-80	blank

The cards are generated in format required for the DNS/Down Translating and Culling Program.

<u>Mode 3</u>

1. First control card in data deck

Col. 1-6	must have *EQUAT
Col. 7-12	blank
Col. 13-18	must have BOTHbb
Col. 19-80	blank

2. Equation cards

3. Immediately following last equation card

Col. 1-6	must have *ENDbE

Cards will be punched for both the DNS/DTC and DNS/Pre-Ed Programs.

METHOD:

The program reads boolean equations from IBM punched cards and culls out duplicate variables and operators. The remaining variables are classified and a specific time and parameter card is punched for use in conjunction with each variable. Initial processing flags are set to zero and data cards are read until the beginning of the equations is sensed. A flag is then set for reference during processing. Format instructions are obtained from the equation beginning control card and a flag is set for program processing direction. The difference is in the placement of time parameters on card image. The Down Translator and Culling Program time field begins in col. C37 while the preprocessor editor requires time cards with the time field starting in col. C13.

Control of processing is transferred to subroutine names. The cards are then read in one at a time, stored, and checked each time for an end of equations indication. Initial entry flag is tested for zero, if so, the working cells are set up for names card processing. The initial entry flag is then turned on, and a card counter is incremented. The first word on each card is sensed for a blank or initial equation card. Only initial equation cards will contain data in word one. Finding data in word one, the program assumes this to be a variable that appears on the left side of the equation and processes it as such. The variable is compared to those stored in a table designated as LNAMS, (left names) to see if it has previously been encountered. If it is a duplicate, it is stored in a separate table and an error count record is incremented. (Names should only appear once on the left hand (LH) side of equation).

The variable is then stored in LNAMS for LH variables and the LH names counter is incremented. The remainder of the card is read and the variables are separated from operators; one variable at a time. As each variable is located it is compared with those in the RH names table. If a variable is found to be a duplicate right name, it is ignored and the next name is obtained from the card. When a period is located or a total of 72 card columns have been tested, a new card is obtained and the same sequence is repeated for each new card until end of equations is sensed. At the completion of names comparison, two tables will have been created.

1. "LH names" variables that appear on the left of an equation with duplicates eliminated and flagged as errors.

2. A list of variables that appear on the right side of the equation with duplicate variables eliminated.

Program control is then transferred to subroutine NARRNG where a second pass through the tables is made, this time comparing the RH table with the left hand table for duplicates. If duplicate variables are found, they are written in a third table and the cells in the LH and RH tables that contained the variables are filled with zeros. After the entire LH and RH tables are processed, the zeroed cells are eliminated by rearranging the variables in each table. Program control shifts to subroutine NSORT and all three tables are then sorted in ascending alpha numeric order and each table classified by variable type.

1. LH variable table contains 'terminals'.

2. Table of variables from LH and RH contains 'transactors'.

3. RH variable table contains 'initiators'.

Each table starting with the 'terminal' table is then processed to identify the following:

1. AMA classification code, (is variable a coil, node, contact, etc.).

2. The time sequencing parameters for each type (node, coil, contact, etc.).

3. Whether format of card is to be in DT&C (extended) or prep-ed (condensed) format.

4. Whether variable is active or inactive (does variable change state during a normal simulation or is its state constant).

Beginning with the 'TERMINAL' table, each variable is processed through subroutine NTYPEZ one at a time. Key words in core are compared with the variable being processed. When a match is found, an integer representing the appropriate code, class or time field is placed in the assignment key KODE.

Control is shifted to subroutine PRINZ where the value of KODE directs the program and selects the correct print and punch format, and this format is then written on a punch tape and printed for reference. The completed punched card will contain the variable name, the time parameters, the active, inactive code, and the AMA class code. The printout will be ordered as follows:

1. List of equations.

2. List of program generated time cards for terminals.

3. List of program generated time cards for transactors.

4. List of program generated time cards for initiators.

5. Number of names processed.

OUTPUT FORMAT:

Examples of typical printout generated by the DNS time card generator is shown in Figures 2-1 through 2-4. The first listing is the total equation cards in the order of input as shown in Figure 2-1. The *END equations card at the bottom of the listing indicates that the entire equation deck was input.

Figure 2-2 is a sample listing of the variables that have been classified as terminals. The sub title on the top left of sheet specifies this fact. To the right of this class sub title is the

```
DNS TIME AND NAME CARD PUNCH PRØGRAM        GENERAL DYNAMICS/CØNVAIR          PAGE NØ   1

                    EQUATIØN LISTING -

DI423     =97K116 */97K119 */96K123 *6D110.
DI424     =97K117 */97K118 */96K123 *6D110.
DI425     =97K118 */97K117 */96K123 *6D110.
DI426     =97K119 */97K116 */96K123 *6D110.
DI427     =DØ968 +97K120 */96K123 *6D110.
DI428     =DØ969 +97K121 */96K123 *6D110.
DI429     =DØ970 +97K122 */96K123 *6D110.
DI498     =NC7A */D7CR5Ø.
DI499     =(/50K26 +1W16J3) *C7K3 */C8KD */C8KC */C7K1A *6D110.
DI500     =/C7K4A */C8K3 *C7KA *C7K1A *6D110.
DI501     =/C7K4A *C8KC */C8K6 *C7K1A *6D110.
DI502     =96K123 *6D110.
DI503     =(/50K26 +1W16J3) *C7K3 *(/C7K1B */C7K2A +C7K2A +/C7K5A +
           C8KD) *6D110.
DI999     = NC7B.
NC7A      =L1NC7A +L2NC7A +L3NC7A +L4NC7A.
NC7B      = L1NC7B +L2NC7B.
6D121     = 6D110 * 7S7ØN.
C7K1A     =/C7K4B *C7K1A *6D110 +C7K1B */C7K2A *6D110.
C7K1B     =DØ972.
C7K1C     =98K115 *6D110.
C7K2A     =C7K2C *C7K1C *6D110.
C7K2B     =DØ973.
C7K2C     =(C7K1C +C75K2 +/C75K2 *C75K3) *6D110.
C7K3      =(C7K1C +C75K2 +/C75K2 *C75K3) *6D110.
C7K4A     =C8KD *6D110.
C7K4B     =(C7K1C +C75K2 +/C75K2 *C75K3) *6D110.
C7K5A     =DØ976.
C7K5B     =C75KF *C8KB *C7K1C *6D110.
L1NC7A    =6D110 *C7KA *(/C7K2A *C7K2B +C7K2A */C7KE +/C8KB).
L2NC7A    =C7K1A */C7K4B *C7K1B */C7K2A *6D110.
*END E
```

Figure 2-1. Equation listing name and time card generator program.

```
DNS TIME AND NAME CARD PUNCH PRØGRAM              GENERAL DYNAMICS/CØNVAIR        PAGE NØ   2

LIST ØF TERMINALS            (      16 VARIABLES )

6D121
6D121                        S    0S    0S    0S    0S    0S    0A

C7K5B
C7K5B                        S    5S    5S    5S    5S    5S    5A    K

DI423
DI423                        S    0S    0S    0S    0S    0S    0A    I

DI424
DI424                        S    0S    0S    0S    0S    0S    0A    I

DI425
DI425                        S    0S    0S    0S    0S    0S    0A    I

DI426
DI426                        S    0S    0S    0S    0S    0S    0A    I

DI427
DI427                        S    0S    0S    0S    0S    0S    0A    I

DI428
DI428                        S    0S    0S    0S    0S    0S    0A    I

DI429
DI429                        S    0S    0S    0S    0S    0S    0A    I

DI498
DI498                        S    0S    0S    0S    0S    0S    0A    I

DI499
```

Figure 2-2. Terminal listing name and time card generator program.

number of variables that were classified as terminals. Each variable is listed two times on the left side of the page. The first printing is made when the punch tape is written for the dictionary card. The second printing is a copy of the time card that will be punched including the time field assigned by the program and the variables classification coding. Printing the variable twice as shown provides a listing where dictionary data may be written for the remaining portion of the cards for keypunching. An example of this application is shown for bus 6D121 in Figure 2-2. The transactors and initiators are listed as shown in Figures 2-3 and 2-4 respectively. Printout for variables in these classifications is the same as previously described for 'terminals'.

After all the cards have been printed an account of names is listed as shown in Figure 2-4.

```
DNS TIME AND NAME CARD PUNCH PRØGRAM          GENERAL DYNAMICS/CØNVAIR      PAGE NØ   3

LIST ØF TRANSACTØRS        (    18 VARIABLES )

C7K1A
C7K1A                      S    5S   5S   5S   5S   5S   5A   K

C7K1B
C7K1B                      S    5S   5S   5S   5S   5S   5A   K

C7K1C
C7K1C                      S    5S   5S   5S   5S   5S   5A   K

C7K2A
C7K2A                      S    5S   5S   5S   5S   5S   5A   K

C7K2B
C7K2B                      S    5S   5S   5S   5S   5S   5A   K

C7K2C
C7K2C                      S    5S   5S   5S   5S   5S   5A   K

C7K3
C7K3                       S    5S   5S   5S   5S   5S   5A   K

C7K4A
C7K4A                      S    5S   5S   5S   5S   5S   5A   K

C7K4B
C7K4B                      S    5S   5S   5S   5S   5S   5A   K

C7K5A
C7K5A                      S    5S   5S   5S   5S   5S   5A   K

L1NC7A
```

Figure 2-3. Transactor listing name and time card generator program.

```
DNS TIME AND NAME CARD PUNCH PRØGRAM              GENERAL DYNAMICS/CØNVAIR      PAGE NØ   5

LIST ØF INITIATØRS           (    35 VARIABLES )

1W16J3
1W16J3                               S    0S    0S    0S    0S    0S    0A

40K6
40K6                                 S    5S    5S    5S    5S    5S    5A    K

50K26
50K26                                S    5S    5S    5S    5S    5S    5A    K

6D110
6D110                                S    0S    0S    0S    0S    0S    0A

7S7ØN
7S7ØN                                S    0S    0S    0S    0S    0S    0A

96K123
96K123                               S    5S    5S    5S    5S    5S    5A    K

97K116
97K116                               S    5S    5S    5S    5S    5S    5A    K

97K117
97K117                               S    5S    5S    5S    5S    5S    5A    K


TØTAL NUMBER ØF NAMES =     69
```

Figure 2-4. Initiator listing and names total, name and time card generator program.

APPENDIX A

PROGRAM FLOW CHARTS

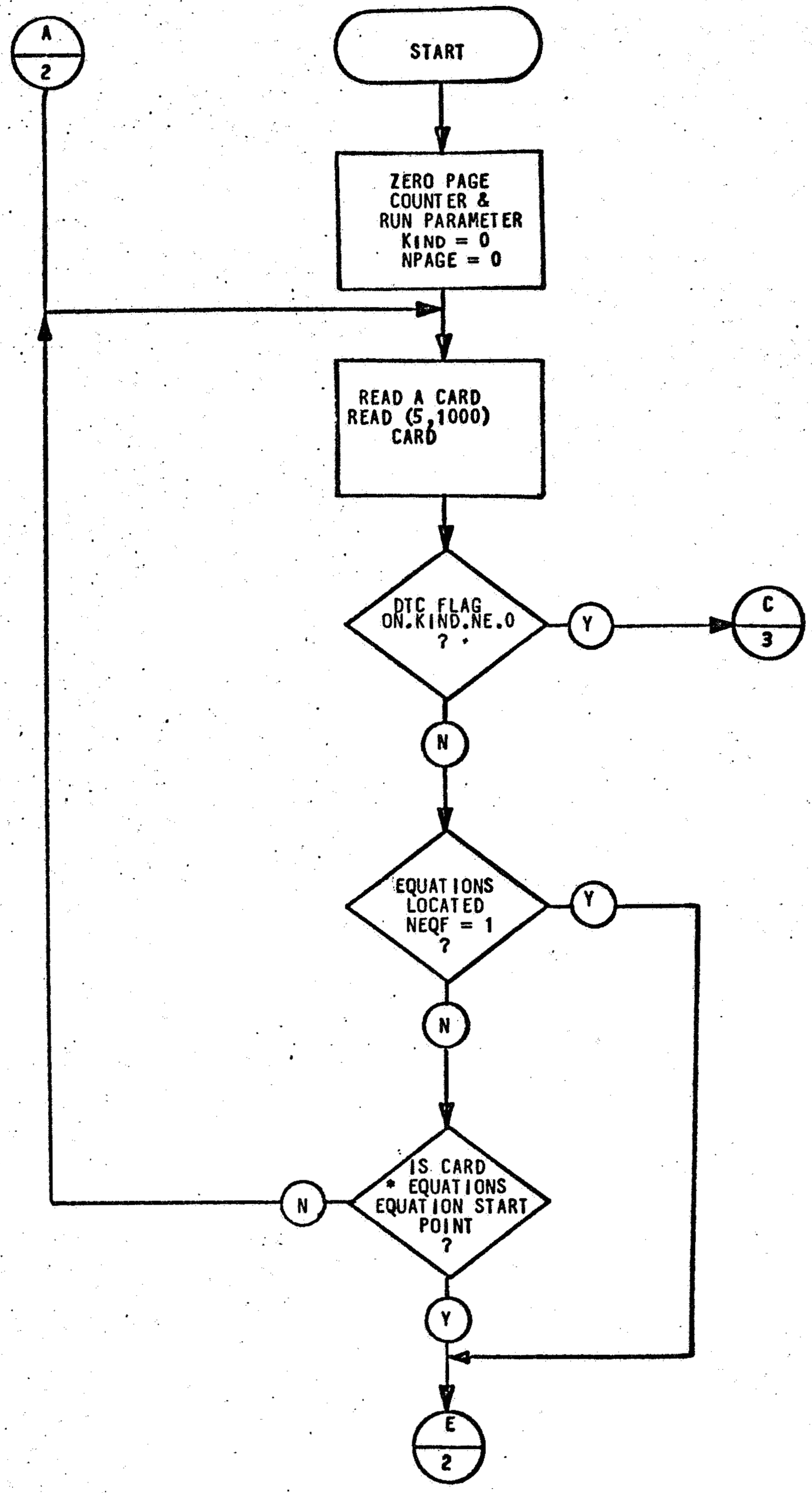


Figure A-1. SUBROUTINE READC (1 of 4)

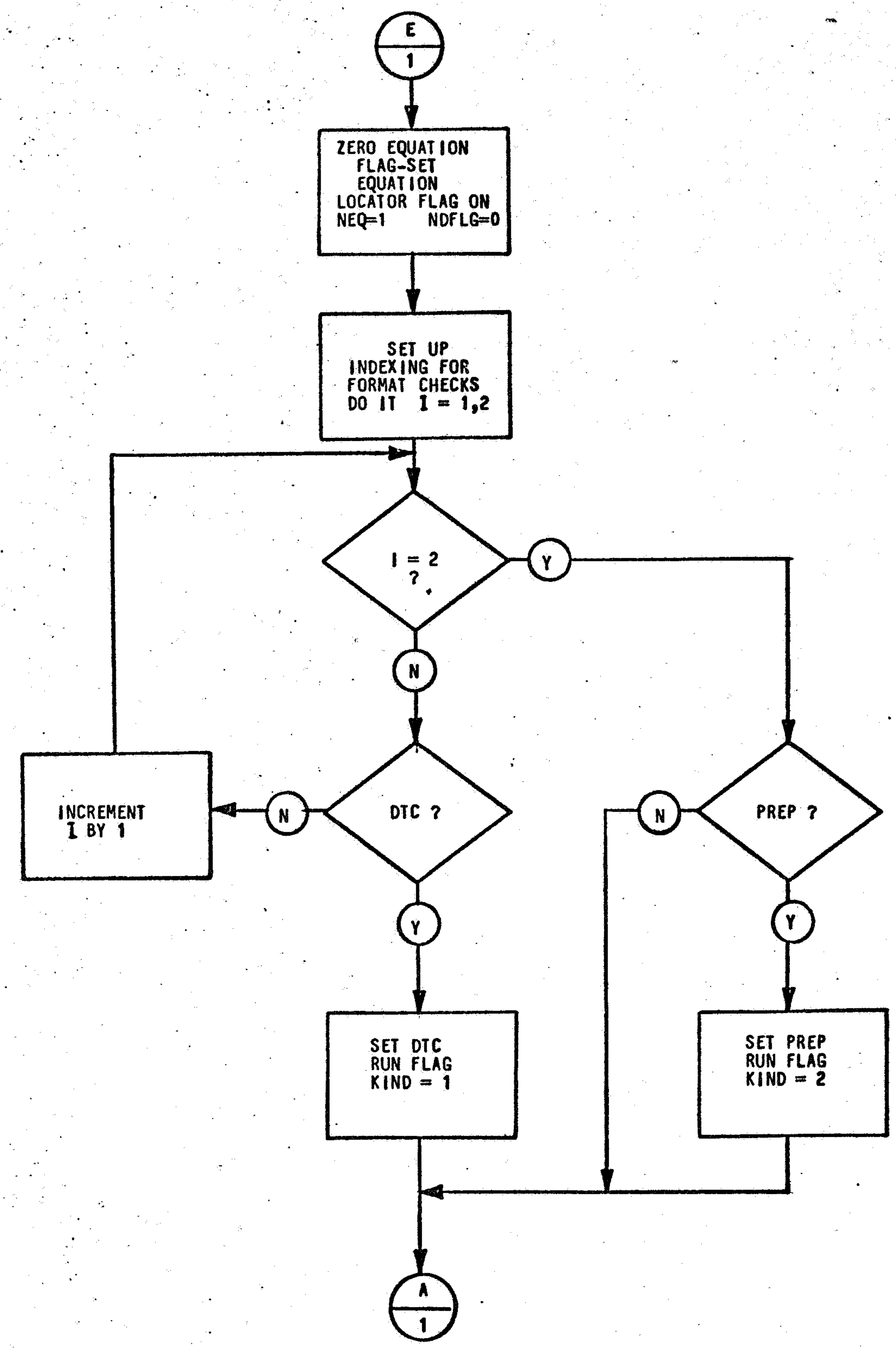

E
1
ZERO EQUATION FLAG-SET EQUATION LOCATOR FLAG ON NEQ=1 NDFLG=0
SET UP INDEXING FOR FORMAT CHECKS DO IT I = 1,2
I = 2 ?
Y
N
DTC ?
N
INCREMENT I BY 1
Y
SET DTC RUN FLAG KIND = 1
PREP ?
N
Y
SET PREP RUN FLAG KIND = 2
A
1

Figure A-1. SUBROUTINE READC (2 of 4)

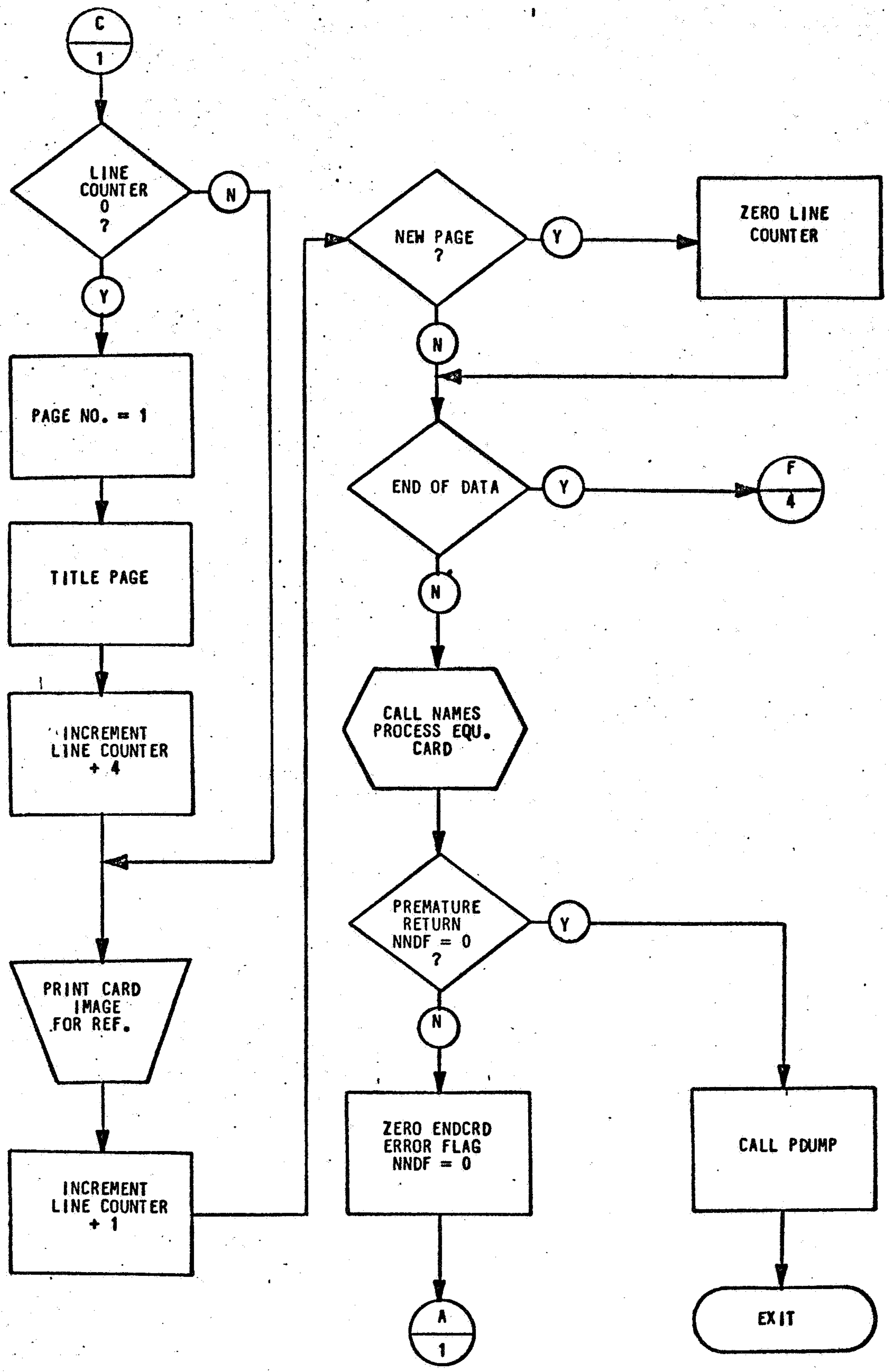

C
1
LINE COUNTER 0 ?
N
Y
PAGE NO. = 1
TITLE PAGE
INCREMENT LINE COUNTER + 4
PRINT CARD IMAGE FOR REF.
INCREMENT LINE COUNTER + 1
NEW PAGE ?
Y
ZERO LINE COUNTER
N
END OF DATA
Y
F
4
N
CALL NAMES PROCESS EQU. CARD
PREMATURE RETURN NNDF = 0 ?
Y
N
ZERO ENDCRD ERROR FLAG NNDF = 0
A
1
CALL PDUMP
EXIT

Figure A-1. SUBROUTINE READC (3 of 4)

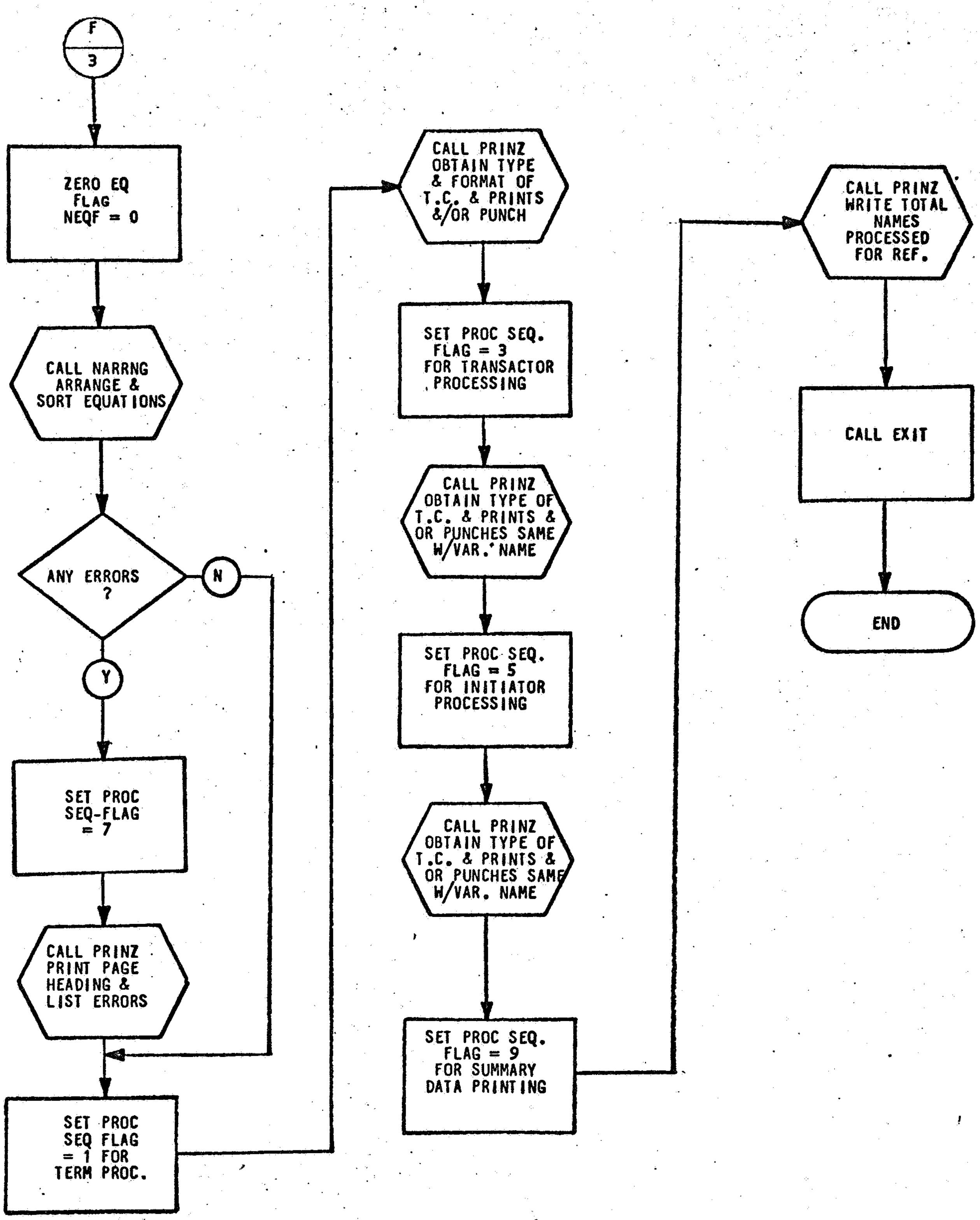
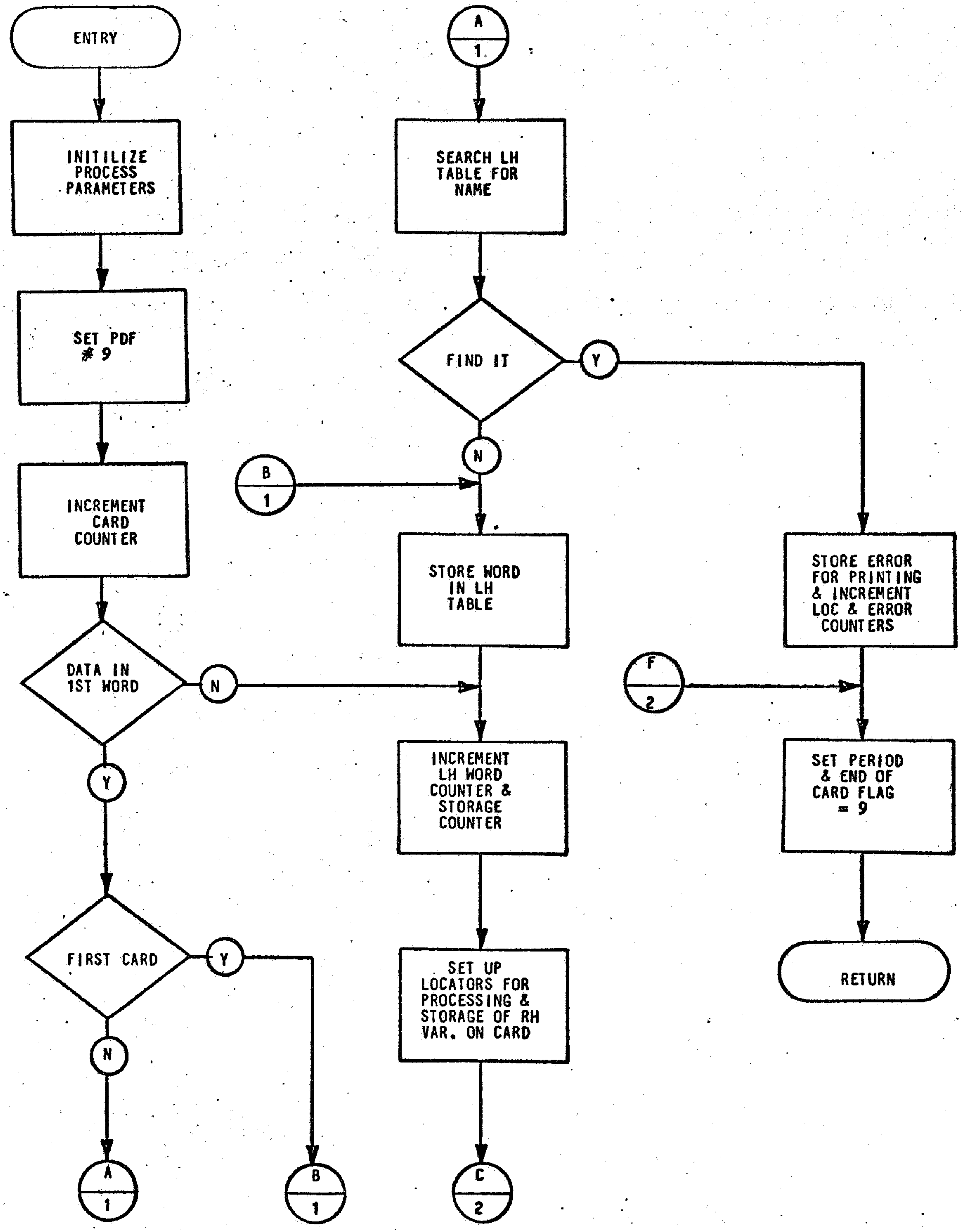

F
3
ZERO EQ FLAG NEQF = 0
CALL NARRNG ARRANGE & SORT EQUATIONS
ANY ERRORS ?
N
Y
SET PROC SEQ-FLAG = 7
CALL PRINZ PRINT PAGE HEADING & LIST ERRORS
SET PROC SEQ FLAG = 1 FOR TERM PROC.
CALL PRINZ OBTAIN TYPE & FORMAT OF T.C. & PRINTS &/OR PUNCH
SET PROC SEQ. FLAG = 3 FOR TRANSACTOR PROCESSING
CALL PRINZ OBTAIN TYPE OF T.C. & PRINTS & OR PUNCHES SAME W/VAR. NAME
SET PROC SEQ. FLAG = 5 FOR INITIATOR PROCESSING
CALL PRINZ OBTAIN TYPE OF T.C. & PRINTS & OR PUNCHES SAME W/VAR. NAME
SET PROC SEQ. FLAG = 9 FOR SUMMARY DATA PRINTING
CALL PRINZ WRITE TOTAL NAMES PROCESSED FOR REF.
CALL EXIT
END

Figure A-1. SUBROUTINE READC (4 of 4)



ENTRY
INITILIZE PROCESS PARAMETERS
SET PDF # 9
INCREMENT CARD COUNTER
DATA IN 1ST WORD
N
Y
FIRST CARD
Y
N
A 1
B 1
A 1
SEARCH LH TABLE FOR NAME
FIND IT
Y
N
B 1
STORE WORD IN LH TABLE
INCREMENT LH WORD COUNTER & STORAGE COUNTER
SET UP LOCATORS FOR PROCESSING & STORAGE OF RH VAR. ON CARD
C 2
STORE ERROR FOR PRINTING & INCREMENT LOC & ERROR COUNTERS
F 2
SET PERIOD & END OF CARD FLAG = 9
RETURN

Figure A-2. SUBROUTINE NAMEZ (1 of 2)

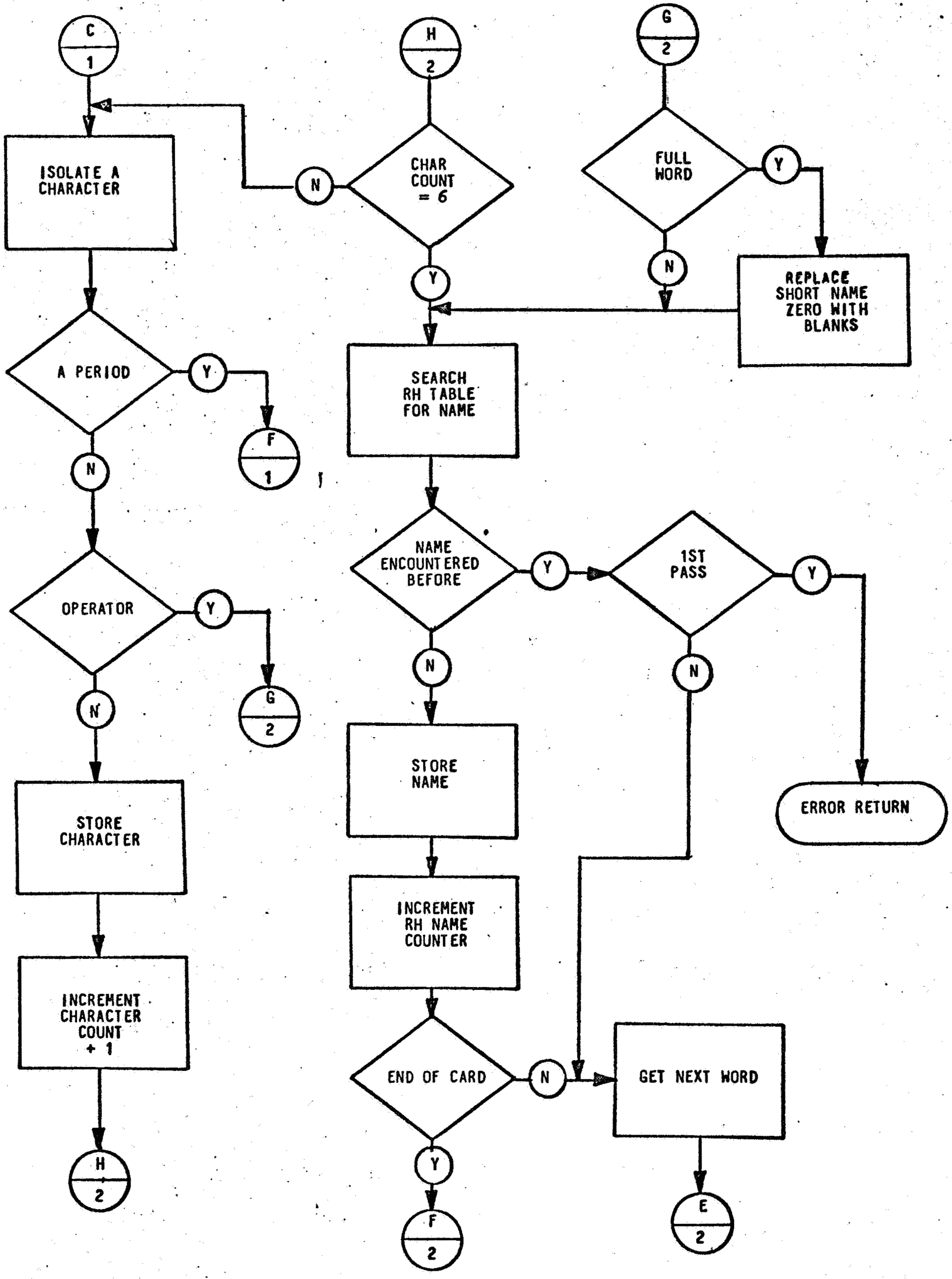


Figure A-2. SUBROUTINE NAMEZ (2 of 2)

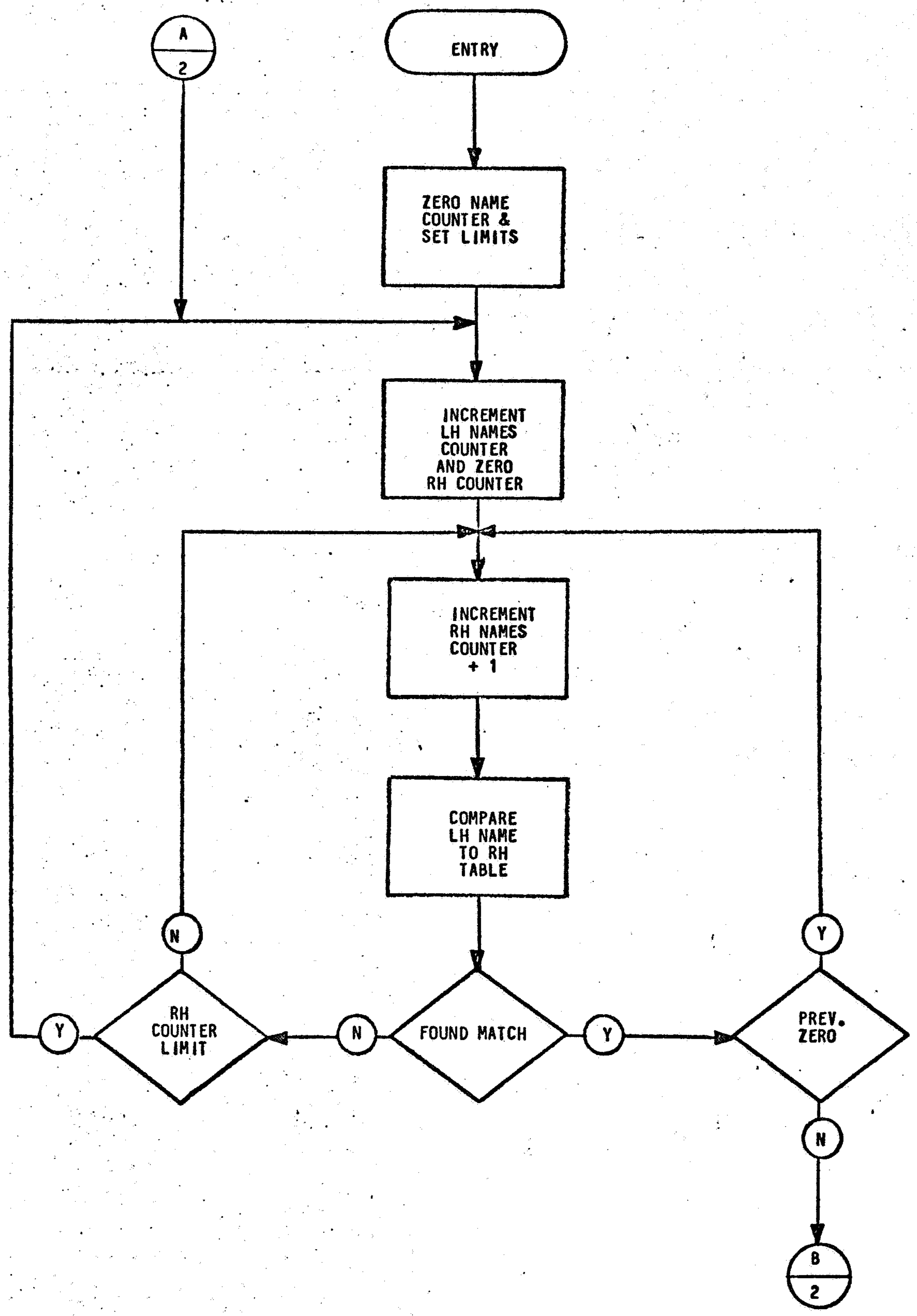

A
2
ENTRY
ZERO NAME COUNTER & SET LIMITS
INCREMENT LH NAMES COUNTER AND ZERO RH COUNTER
INCREMENT RH NAMES COUNTER + 1
COMPARE LH NAME TO RH TABLE
N
Y
RH COUNTER LIMIT
N
FOUND MATCH
Y
PREV. ZERO
N
B
2

Figure A-3. SUBROUTINE NARRNZ (1 of 5)

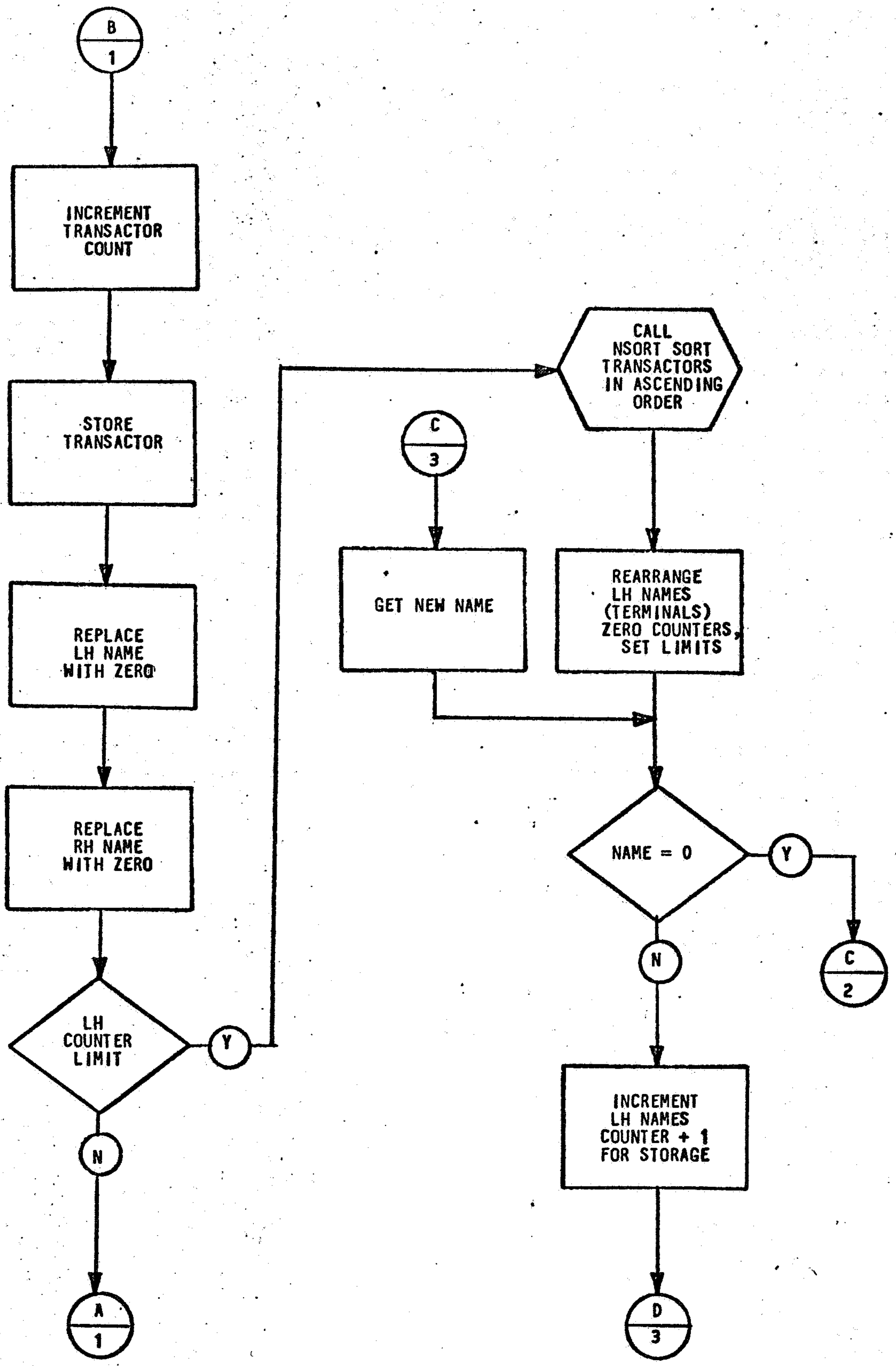

B 1
INCREMENT TRANSACTOR COUNT
STORE TRANSACTOR
REPLACE LH NAME WITH ZERO
REPLACE RH NAME WITH ZERO
LH COUNTER LIMIT
Y
N
A 1
CALL NSORT SORT TRANSACTORS IN ASCENDING ORDER
C 3
GET NEW NAME
REARRANGE LH NAMES (TERMINALS) ZERO COUNTERS, SET LIMITS
NAME = 0
Y
C 2
N
INCREMENT LH NAMES COUNTER + 1 FOR STORAGE
D 3

Figure A-3. SUBROUTINE NARRNZ (2 of 5)

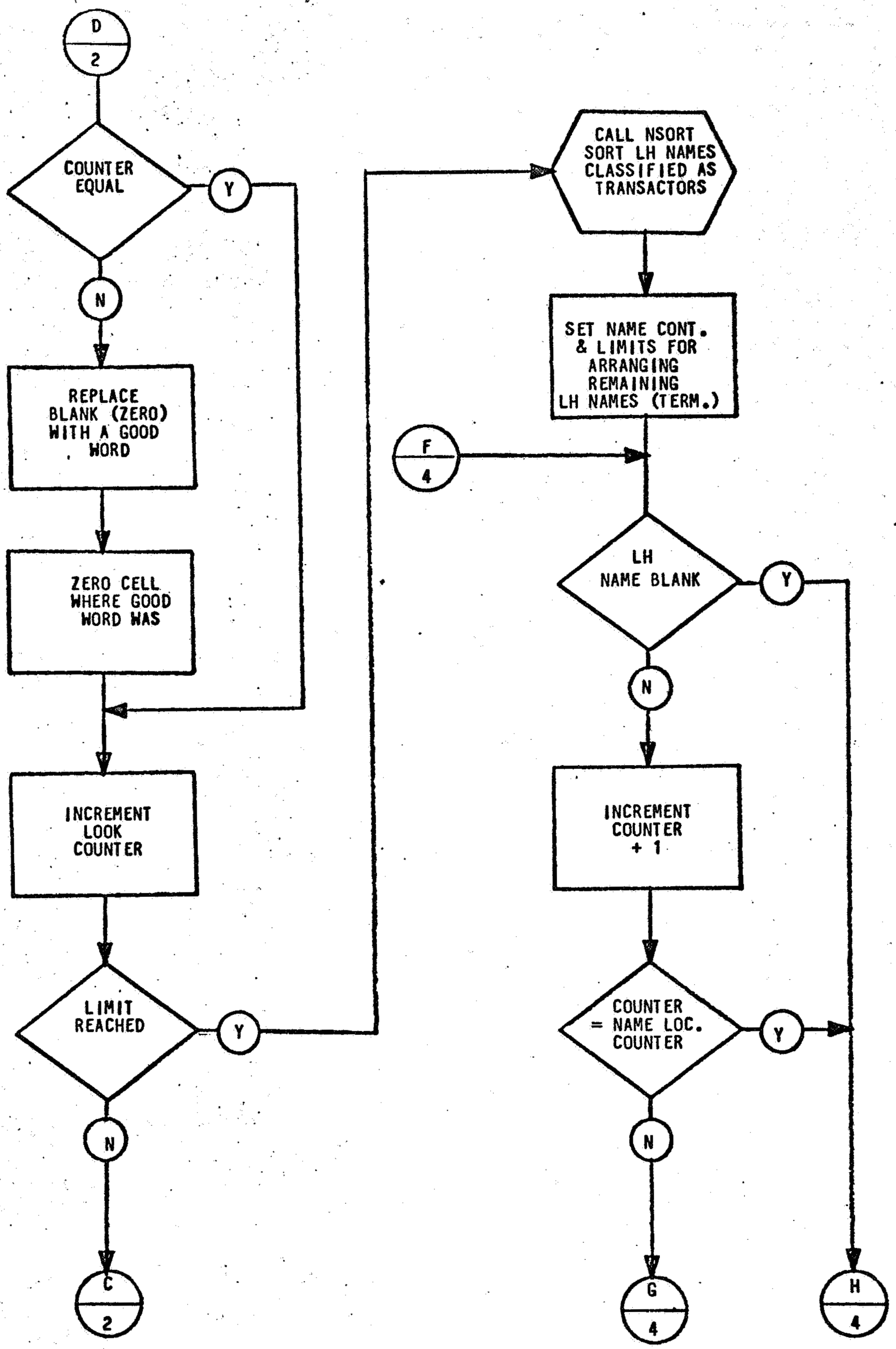

D
2
COUNTER EQUAL
Y
N
REPLACE BLANK (ZERO) WITH A GOOD WORD
ZERO CELL WHERE GOOD WORD WAS
INCREMENT LOOK COUNTER
LIMIT REACHED
Y
N
C
2
CALL NSORT SORT LH NAMES CLASSIFIED AS TRANSACTORS
SET NAME CONT. & LIMITS FOR ARRANGING REMAINING LH NAMES (TERM.)
F
4
LH NAME BLANK
Y
N
INCREMENT COUNTER + 1
COUNTER = NAME LOC. COUNTER
Y
N
G
4
H
4

Figure A-3. SUBROUTINE NARRNZ (3 of 5)

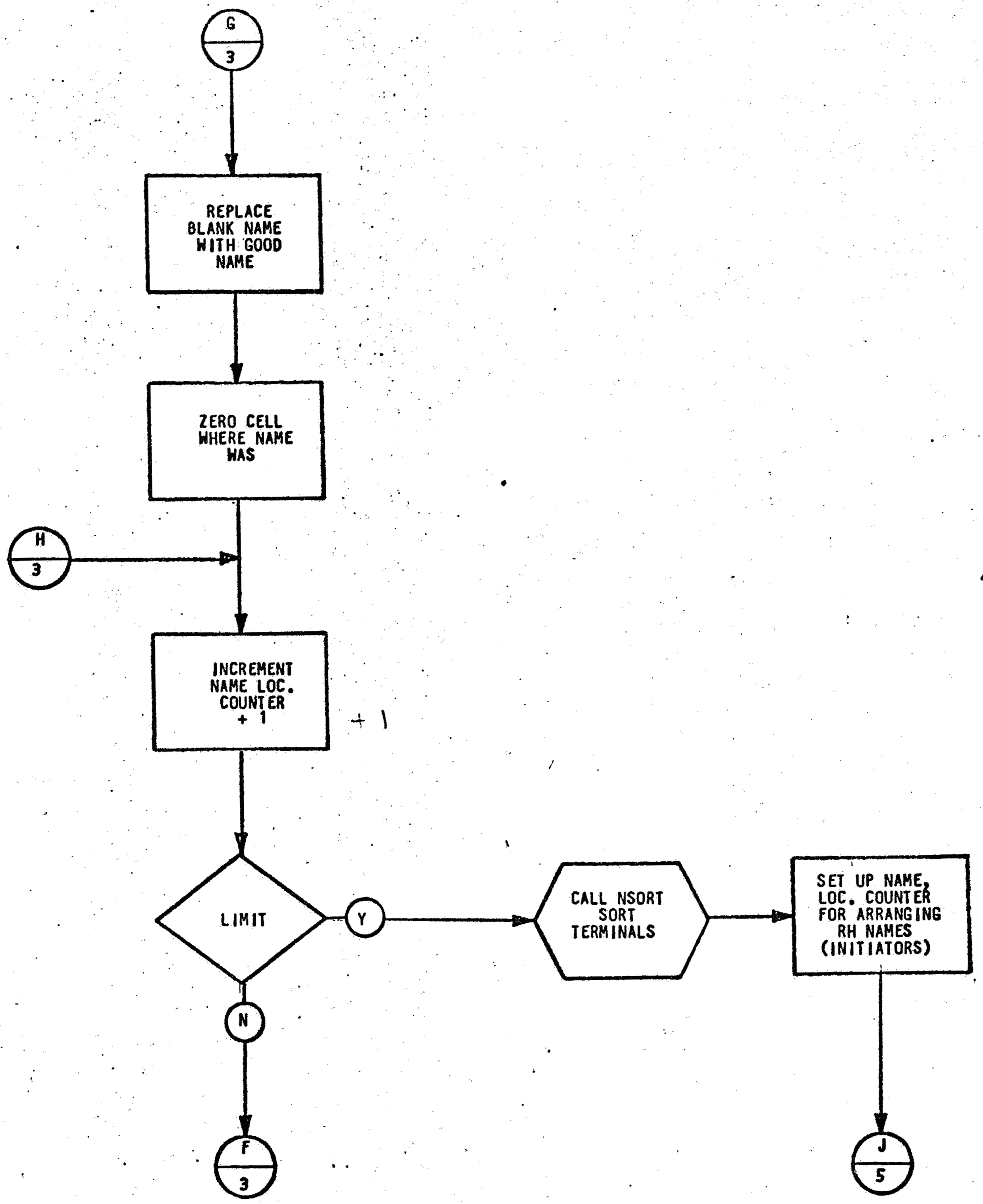

G
3
REPLACE BLANK NAME WITH GOOD NAME
ZERO CELL WHERE NAME WAS
H
3
INCREMENT NAME LOC. COUNTER + 1
+ 1
LIMIT
Y
N
CALL NSORT SORT TERMINALS
SET UP NAME, LOC. COUNTER FOR ARRANGING RH NAMES (INITIATORS)
F
3
J
5

Figure A-3. SUBROUTINE NARRNZ (4 of 5)

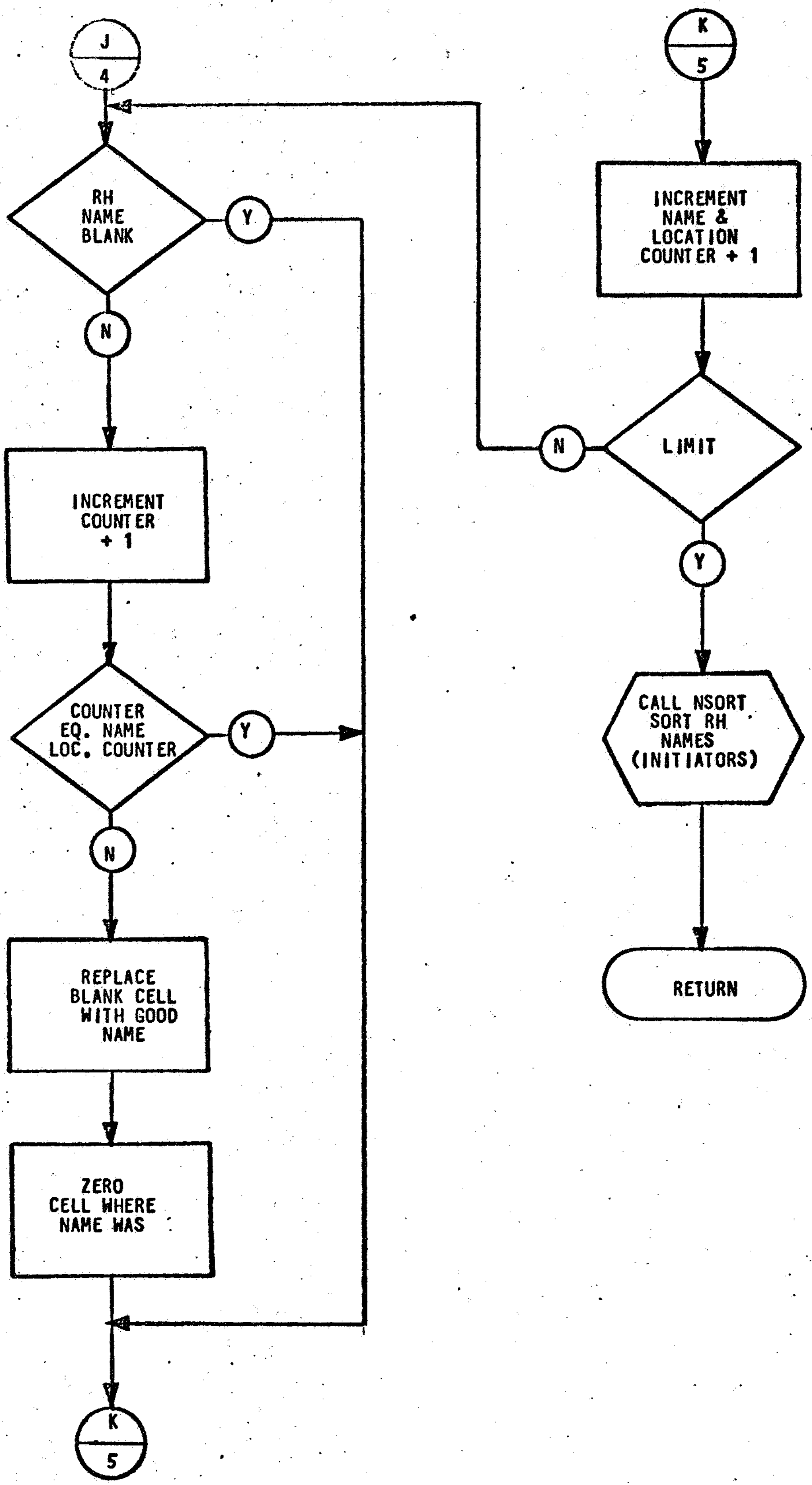

J
4
RH NAME BLANK
Y
N
INCREMENT COUNTER + 1
COUNTER EQ. NAME LOC. COUNTER
Y
N
REPLACE BLANK CELL WITH GOOD NAME
ZERO CELL WHERE NAME WAS
K
5
K
5
INCREMENT NAME & LOCATION COUNTER + 1
LIMIT
N
Y
CALL NSORT SORT RH NAMES (INITIATORS)
RETURN

Figure A-3. SUBROUTINE NARRNZ (5 of 5)

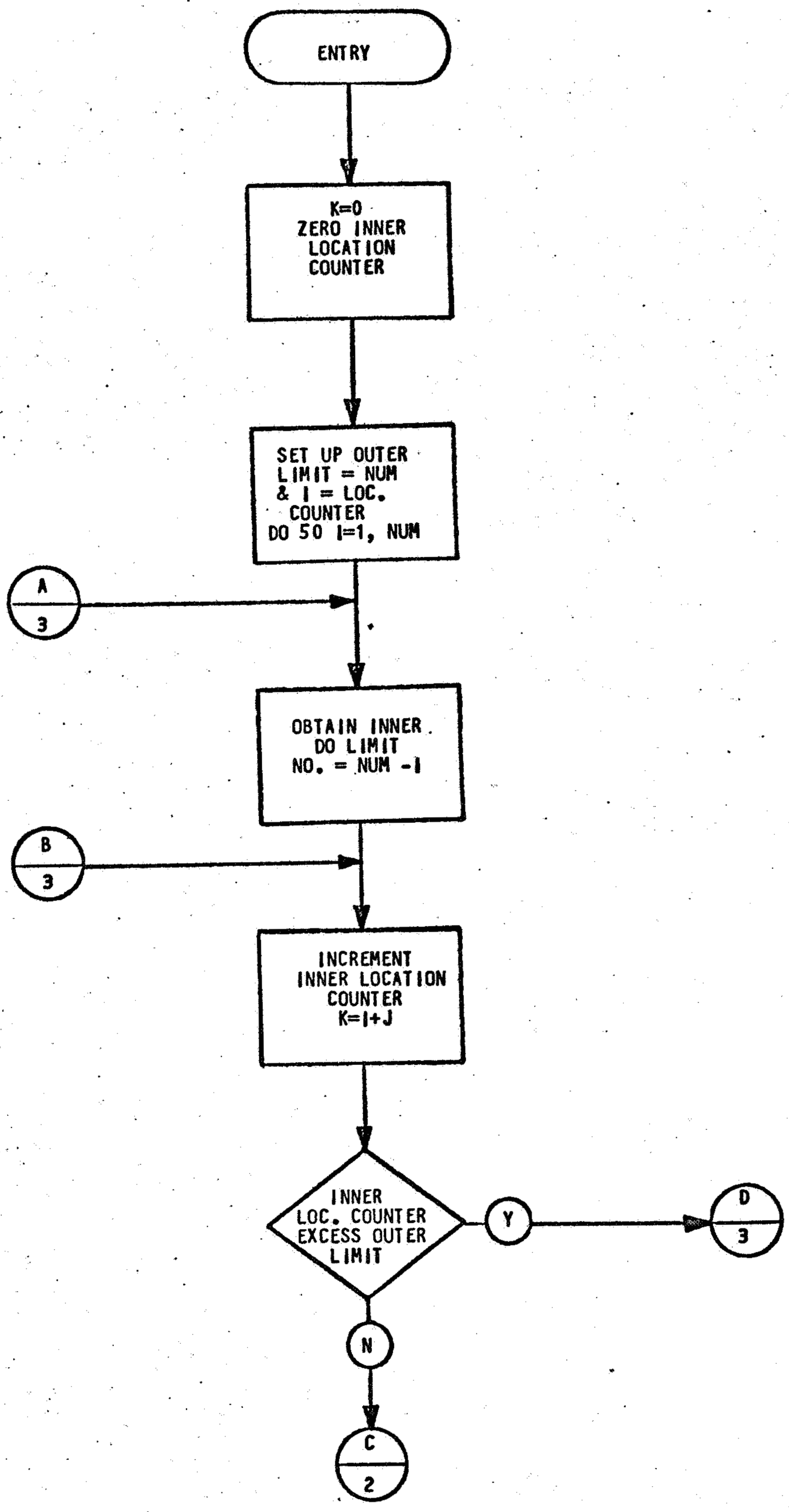


Figure A-4. SUBROUTINE NSORTZ (1 of 3)

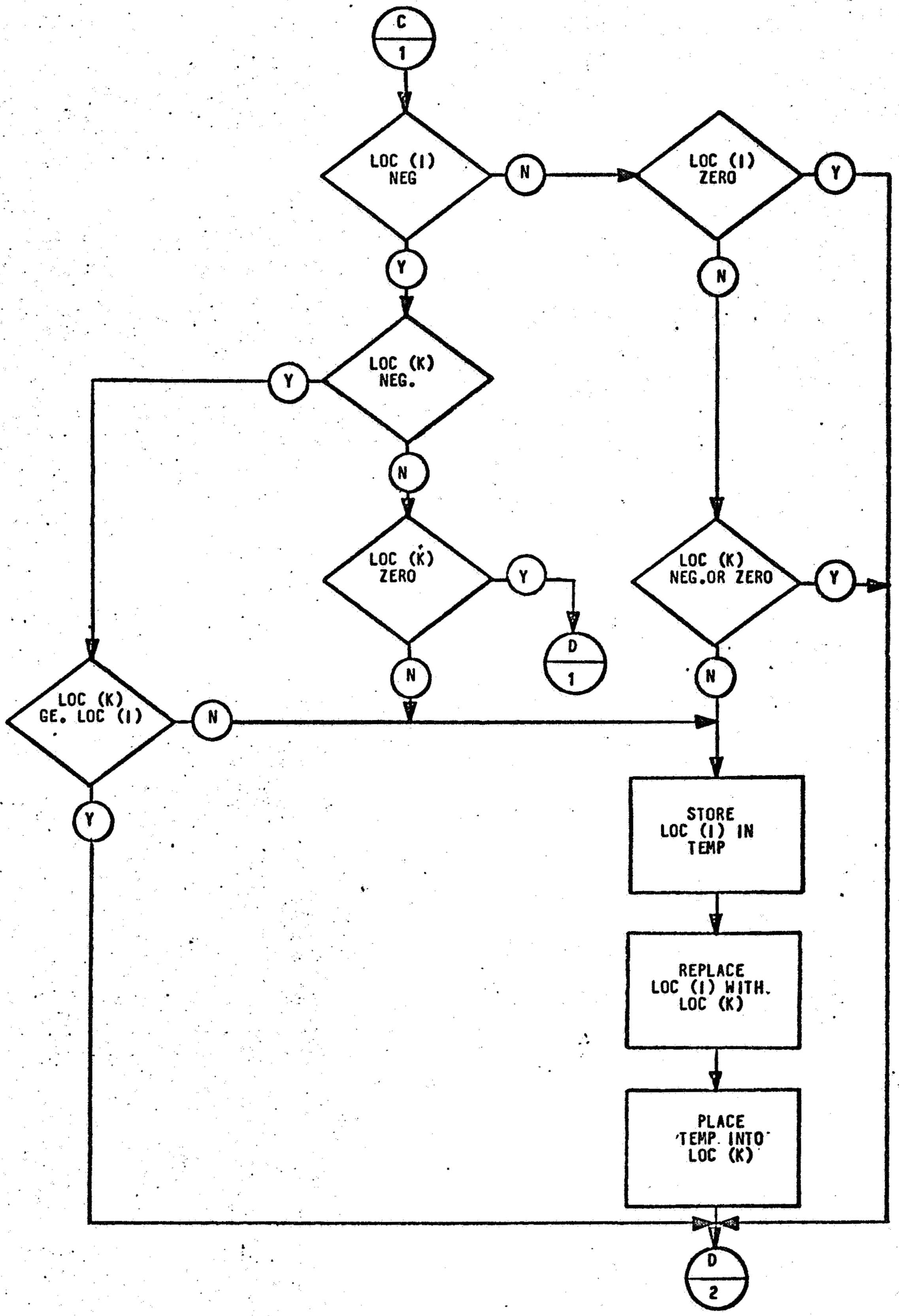


Figure A-4. SUBROUTINE NSORTZ (2 of 3)

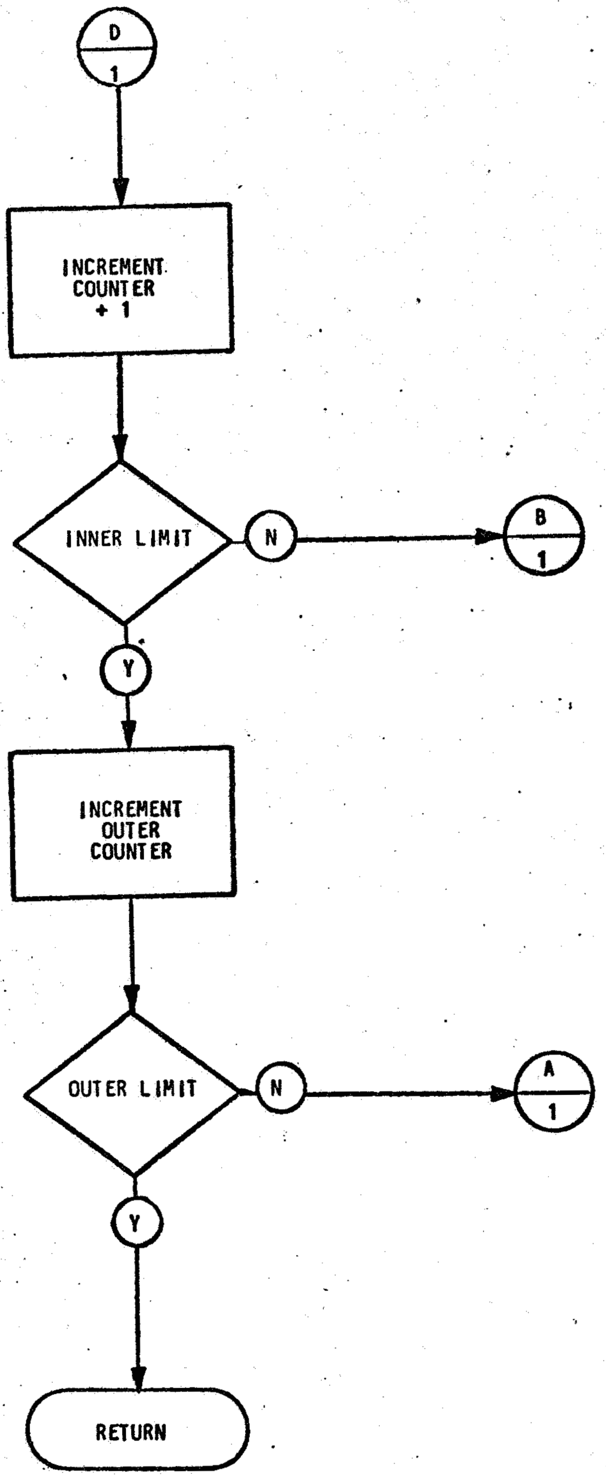


Figure A-4. SUBROUTINE NSORTZ (3 of 3)

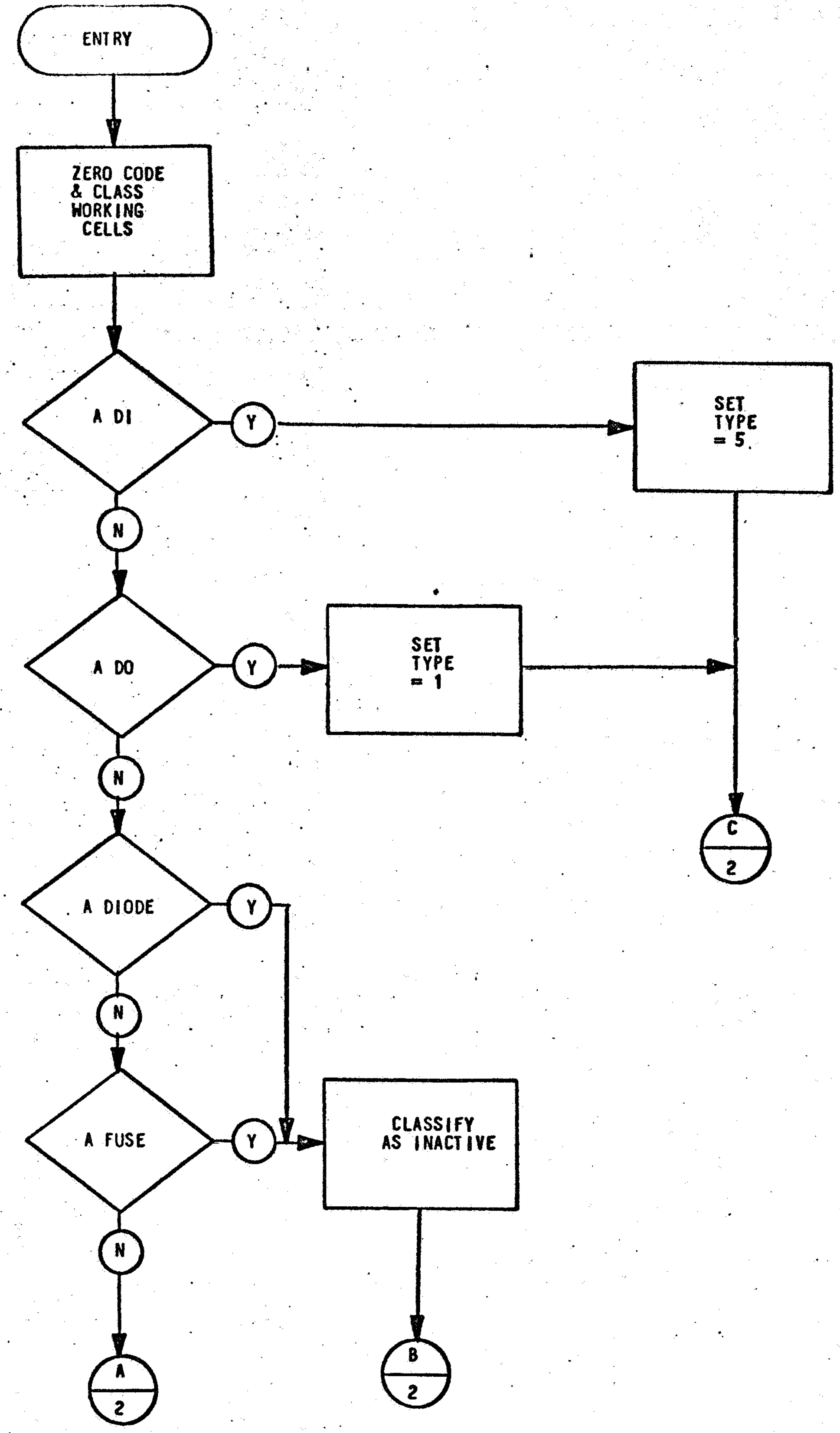


Figure A-5. SUBROUTINE NTYPEZ (1 of 2)

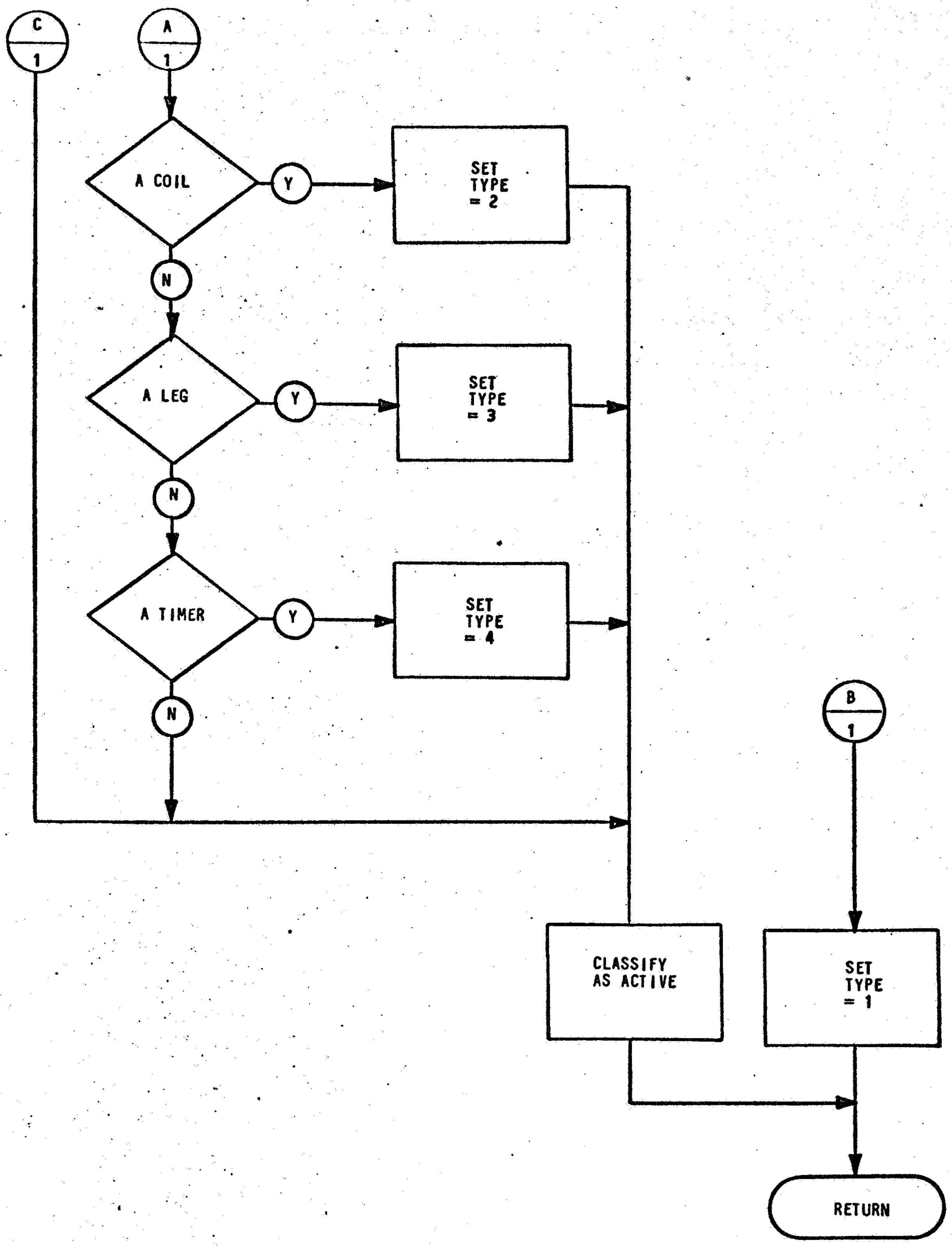


Figure A-5. SUBROUTINE NTYPEZ (2 of 2)

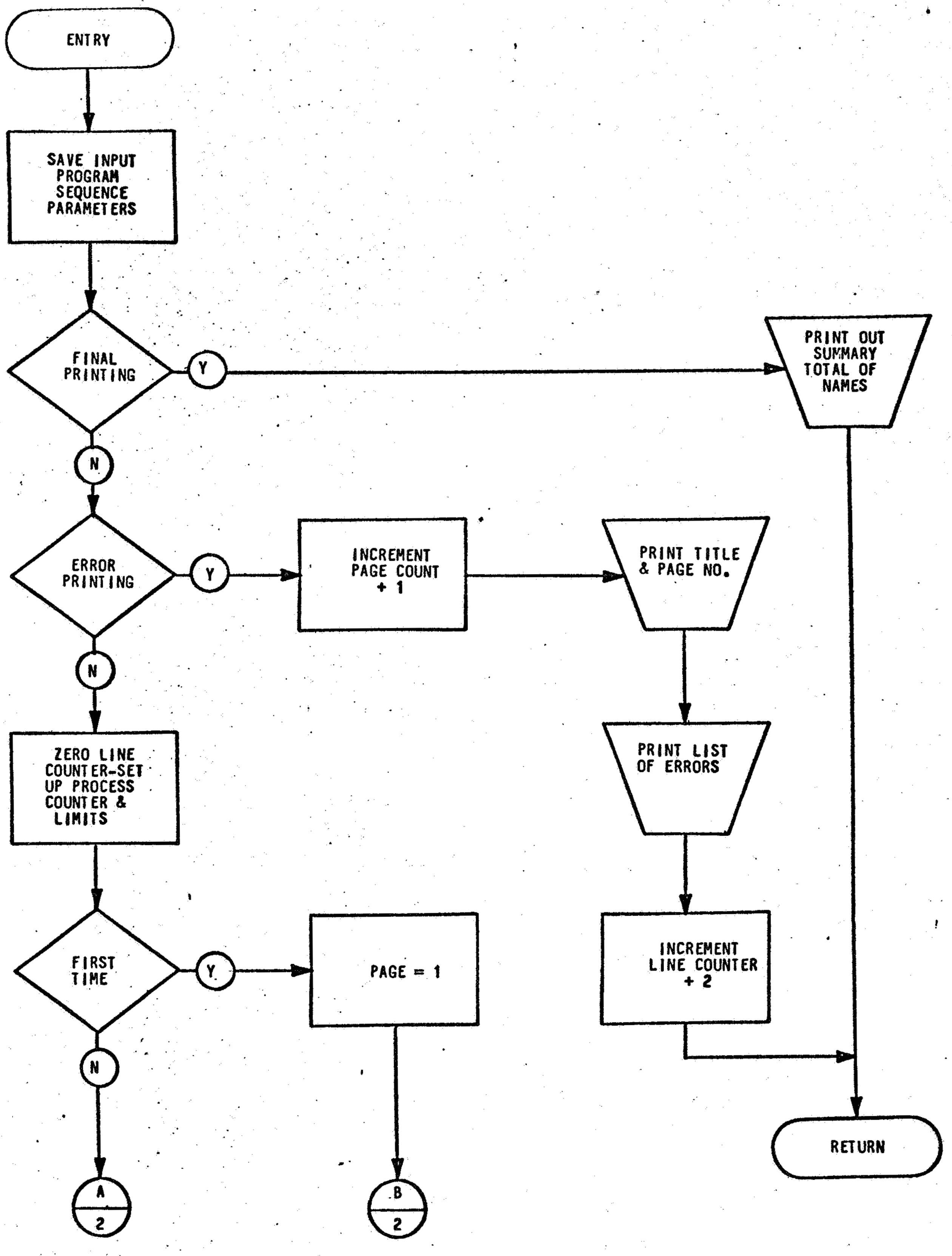

ENTRY
SAVE INPUT PROGRAM SEQUENCE PARAMETERS
FINAL PRINTING
Y
PRINT OUT SUMMARY TOTAL OF NAMES
N
ERROR PRINTING
Y
INCREMENT PAGE COUNT + 1
PRINT TITLE & PAGE NO.
N
ZERO LINE COUNTER-SET UP PROCESS COUNTER & LIMITS
PRINT LIST OF ERRORS
FIRST TIME
Y
PAGE = 1
INCREMENT LINE COUNTER + 2
N
RETURN
A 2
B 2

Figure A-6. SUBROUTINE PRINX (1 of 4)

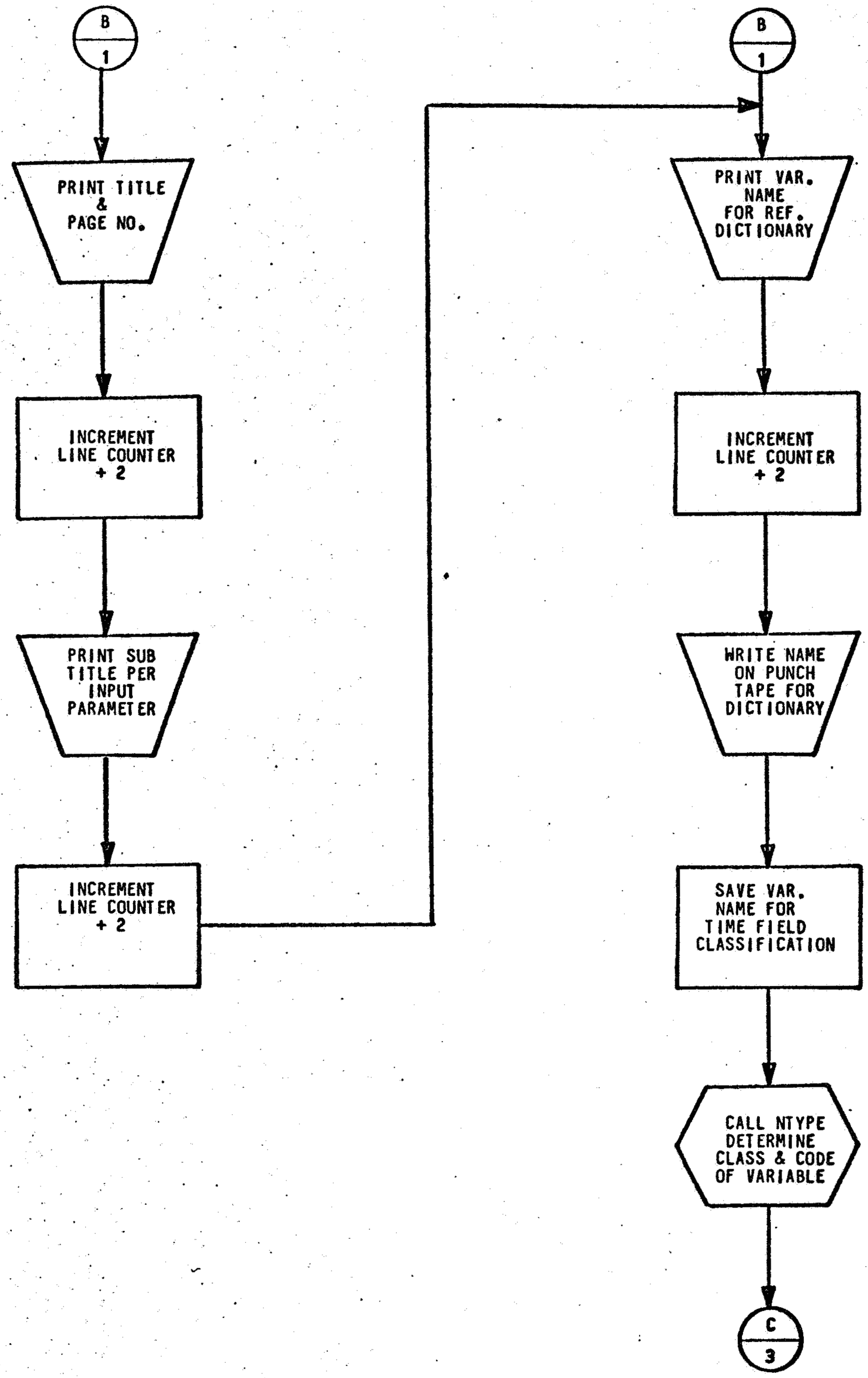

B
1
PRINT TITLE & PAGE NO.
INCREMENT LINE COUNTER + 2
PRINT SUB TITLE PER INPUT PARAMETER
INCREMENT LINE COUNTER + 2
B
1
PRINT VAR. NAME FOR REF. DICTIONARY
INCREMENT LINE COUNTER + 2
WRITE NAME ON PUNCH TAPE FOR DICTIONARY
SAVE VAR. NAME FOR TIME FIELD CLASSIFICATION
CALL NTYPE DETERMINE CLASS & CODE OF VARIABLE
C
3

Figure A-6. SUBROUTINE PRINX (2 of 4)

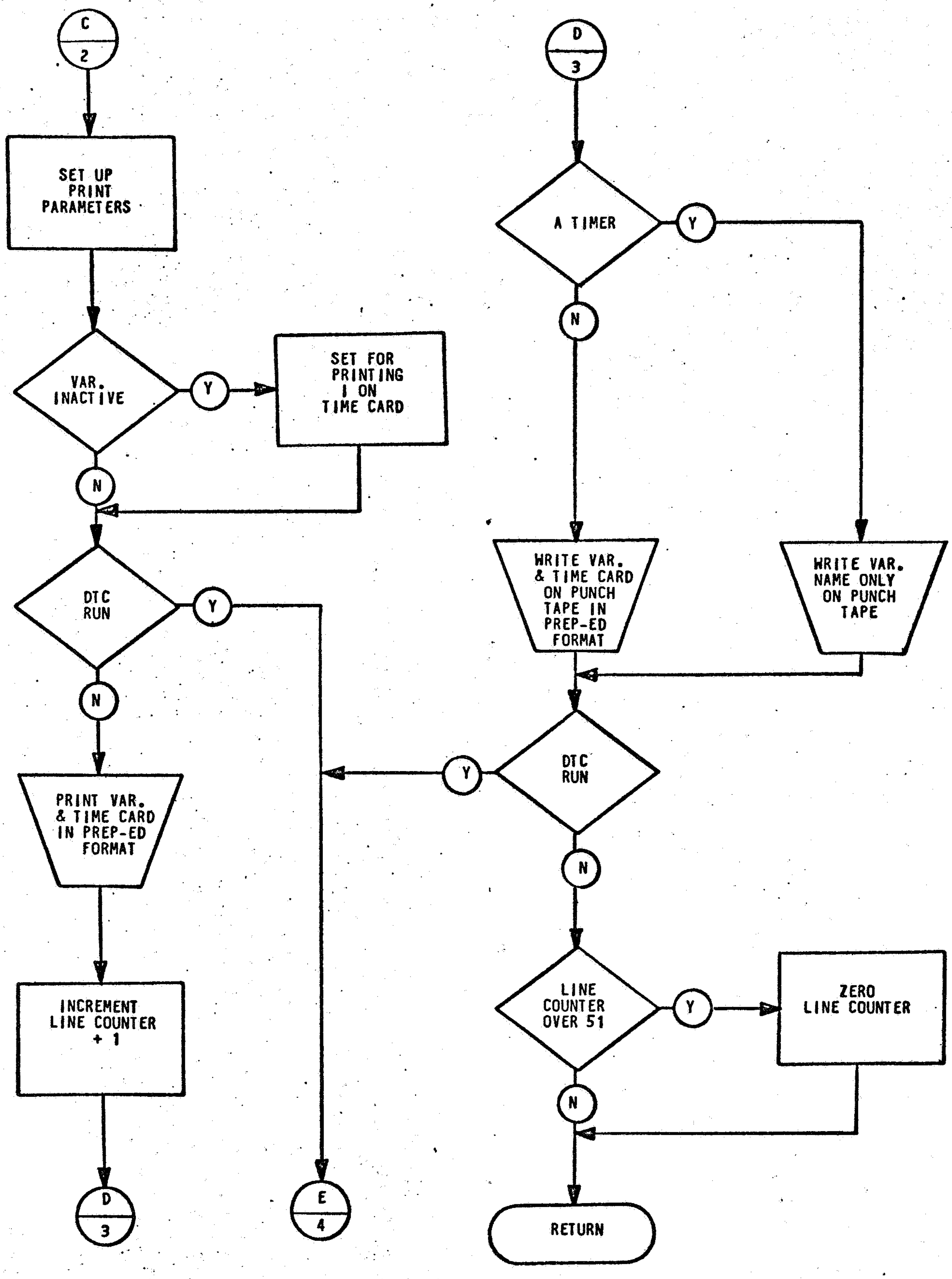

C 2
SET UP PRINT PARAMETERS
VAR. INACTIVE
Y
SET FOR PRINTING I ON TIME CARD
N
DTC RUN
Y
N
PRINT VAR. & TIME CARD IN PREP-ED FORMAT
INCREMENT LINE COUNTER + 1
D 3
E 4
D 3
A TIMER
Y
N
WRITE VAR. & TIME CARD ON PUNCH TAPE IN PREP-ED FORMAT
WRITE VAR. NAME ONLY ON PUNCH TAPE
DTC RUN
Y
N
LINE COUNTER OVER 51
Y
ZERO LINE COUNTER
N
RETURN

Figure A-6. SUBROUTINE PRINX (3 of 4)

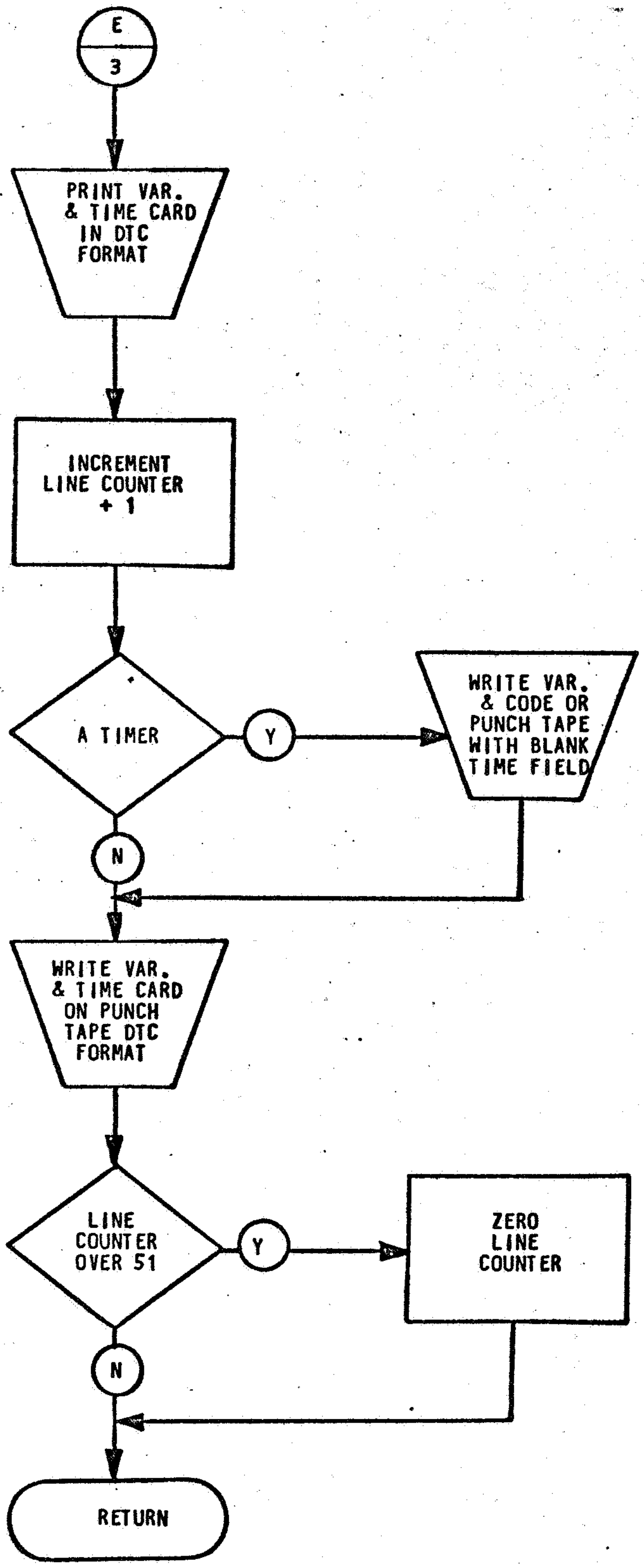

E
3
PRINT VAR. & TIME CARD IN DTC FORMAT
INCREMENT LINE COUNTER + 1
A TIMER
Y
WRITE VAR. & CODE OR PUNCH TAPE WITH BLANK TIME FIELD
N
WRITE VAR. & TIME CARD ON PUNCH TAPE DTC FORMAT
LINE COUNTER OVER 51
Y
ZERO LINE COUNTER
N
RETURN

Figure A-6. SUBROUTINE PRINX (4 of 4)

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLES

The following is an alphabetical listing of the terms used in the NAME/TIME Card Generator Program.

NAME	DESCRIPTION
INAC	Status flag.
KARD	Card input buffer.
KENDS	Storage block end.
KIND	Time card format flag.
KODE	Time parameters key.
LCARD	Storage cell.
LHE	LH names count.
LHERR	Storage cell.
NAMCT	Names count.
NAME	Storage block.
NCOM	Transactor count.
NDFG	Process start flag.
NLIKE	Storage block.
NNDF	Name end flag.
NPAG	Page count.
NTES	Name in process.
IND	Transfer indication.
NCT	Print line count.
NEQF	Equations in process.

2.0 DEFINITIONS

NAME	DESCRIPTION
INAC	Used as a flag to signal variables whose time cards are to be coded inactive. The variable name is tested for presence of specified hardware code letters. If a match is found, the flag is set to one. For all other hardware codes, the flag is zeroed.

NAME	DESCRIPTION
KARD	Fourteen word input buffer for temporary storage of data read from the control and equation cards.
KENDS	Not used.
KIND	Used as a flag to signal the type and format to be used in preparing time cards. If time cards are to be used directly with the Preprocessor Program, KIND is set to zero. If time cards are to be used with the DT&C program, KIND is set to one. If individual time cards are to be prepared for use with both the DT&C and Preprocessor Programs, KIND is set to two. Value to be assigned is signaled by keywords on the *EQUATION control card.
KODE	Used as a flag to signal which hardware code and which group of prespecified pickup and dropout times are to be assigned to a variable's time card. The variable name is tested for the presence of prespecified hardware type code letter. If a match is found, the flag is set to a value corresponding to the particular code letter. If no match is found, the flag is set to zero.
LCARD	Used to store the sequence number (position of the card in the equation card deck) of an equation card which was found to contain a left hand name that had been encountered previously.
LHE	Counter for counting and storing.
LHERR	Storage array for storing up to eight (8) left hand names, should names be inadvertently duplicated or mispelled.
NAMCT	Multipurpose counter used first to count the number of unique names encountered on the RH sides of the equations, second to count the number of initiators, and third to store the total number of names processed.
NAME	Storage array for storing initially the unique list of names encountered on the RH sides of the equations, and later to store the list of names found to be initiators.

NAME	DESCRIPTION
NCOM	Common location for storing the total number of variables found to be transactors.
NLIKE	Storage array for storing the list of names of variables found to be transactors.
NDFG	Entry flag used after initial entry to signal subsequent entries that storage registers and counters have already been initialized.
NNDF	Flag to signal end of current equation card field (either 72 or a period) has been encountered.
NEQF	Flag used to signal that the equation card deck is being read and processed. The flag is set to 1 when the '*EQUATIONS' control card is encountered, and re-zeroed when the '*END EQUATIONS' control card is encountered.

SECTION

3

DNS/ATOLL INPUT CONVERSION AND PUNCH PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

3

DNS/ATOLL INPUT CONVERSION AND PUNCH PROGRAM

AUTHOR:	A. R. Stone Convair division of General Dynamics
PURPOSE:	To utilize the Discrete Network Simulation (DNS) Programs for test procedure verification, it is necessary to derive driving functions for use with the Simulation Program from the ATOLL test procedure itself. This program uses the following techniques to simplify the generation of input data. 1. Read the ATOLL test tape and identify all necessary data required for DNS input. 2. Convert this data into a format that will be compatible with DNS. 3. Punch IBM cards containing the DNS driving commands. 4. Provide necessary written instructions to insure manual inputs are accounted for in DNS simulation.
RESTRICTIONS:	1. The program must run on an IBM 7094 with IBJOB systems capability. 2. In addition to system input and output, two magnetic tape units are required for BCD tapes. 3. A maximum of ten tests may be processed during one computer run. 4. The program was designed for use with ATOLL card image provided by Boeing or IBM.

STORAGE: The program, including files will start at location 02720 and continue through location 23253. It contains 8 subroutines as follows:

1.	CONTRD	Driver
2.	KATLT	General ATOLL card image processing
3.	KATRG	Variable and case, identification
4.	KONVRT	ATOLL step identification
5.	NBRANZ	Test procedure branch identification
6.	NIDEX	IDA No. conversion
7.	READCD	Control card reading and interpretation
8.	TIMEZ	Processing of time field on ATOLL tape

TIMING: Program will generate input data for the DNS Simulation Program at approximately 1000 lines per minute. Each line (or card) represents either an input equation or a control card. The output per minute increases as the number of inputs associated with an ATOLL instruction increases.

USE: A typical operation deck set up for using this program would be as follows:

$JOB
Binary program deck
$DATA
Job specification card
7/8 (EOF)

Job specification card

Col. 1-2	Zero or blank
Col. 3	Zero or integer 1 (1, scan times to be used)
Col. 4-5	Zero or blank
Col. 6	Integers 1 or 2 (1 = IBM, 2 = Boeing)
Col. 7-8	Zero or blank
Col. 9	Integer 2 or blank (2, List on Save Tape)
Col. 10-12	Number of IDA's to be processed - right adjusted to Col. 12.
Col. 13-72	IDA numbers prefixed with "DA" from IDA e.g. DA5002 or blank.

Tapes Requirements

Fortran Logical	System Function	Tape
8	A5	ATOLL test proc.
11	B6	BCD Save (in lieu of punch cards)

METHOD:

The input conversion and punch program processes ATOLL test procedure card image tape records. The records are read and processed case by case until an end of record is encountered. The first case consists of the data starting with the initial ATOLL input (DISO or SEMI) operator encountered, and ending with the first succeeding ATOLL "Scan" operator. Each succeeding case starts with the first input (DISO or SEMI) encountered after a preceeding "Scan" operator, and ends with the next "Scan" operator encountered. The combined ATOLL step and substep numbers at which each new "first input" occurs become the unique identifiers for signalling the start and end of each case in creating the DNS inputs. (In the succeeding comparator program, these unique step-substep numbers are used to correlate the simulation results with the ATOLL predictions on this tape). Any input data contained between these "case controls" is identified, converted, and sequenced to create the inputs for driving or stimulating the DNS model.

1. The control card is first read to determine the format of the ATOLL input tape (IBM or Boeing). The type of output to be created (tape or punched cards), and the number and names of the test procedures to be processed.

2. The "IDA's yet to be processed" counters and control flags are set up. If IDA's remain to be processed, proceeds to 3. If no more IDA's left, cleans up and exits from the machine.

3. The file flag and entry flags are reset for the start of the test procedure.

4. Each ATOLL record (14 word) is read, one record at a time. If the "file found" flag has been set, the processing skips to 5.

A. Checks for current IDA number or name in IBM or Boeing format depending on setting of "input format flag". When correct IDA is encountered, a "file found" flag is set to bypass this portion of the program, and a "control flag" is set to normal value. If correct IDA number is not found in ten names records, the "control flag" is set to "error value", and processing is terminated.

5. If IBM format, skips to 6.

   A. If Boeing format, checks for presence of a block operator. If none, skips to 6.

   B. If a block operator is found, processes current block number as set by "output list" flag, and returns to 4.

6. Checks for presence of an END operator. If none, skips to 7.

   A. If an END operator is found, processes the END record as set by the "output list" flag, and cleans up for return to 2.

7. Checks for presence of any one of a preset list of ATOLL operators. If one is found, skips to 8.

   A. If miscellaneous operator, checks and updates step-substep number if required, and returns to 4.

8. Updates the step-substep number and proceeds to appropriate sections as follows:

   A. If ATOLL operator is a DISO, goes to 9.
      SEMI, goes to 12.
      TEST, goes to 13.
      SCAN, goes to 11.

9. If ATOLL operator was a "DIS01" or "DISO0", sets a value flag and checks to see if this is the first new input encountered.

   A. If not the first, skips to 9C.

B. If first new input, sets "new start" flag on and processes a new DNS *STEP record for this step-substep number.

C. If "DISO input" flag on (a "DISO" has been encountered before) skips to 9E.

D. If this was first new DISO operator, processes new DNS clock input time for start of case, and sets "DISO input" flag.

E. If no time field data encountered, zero's time flag and skips to 10.

F. If timing data encountered, sets "time flag", converts time to binary or storage, and proceeds to 10.

10. Identifies the discrete outs listed in the variables field, checks value flag, and formats and stores the DISO names, values, and current DNS clock time in print storage. If "time flag" is off, skips to 10B.

A. If "time flag" is one, also stores DISO names and opposite values in temporary storage.

B. If end of data or end of variables field, returns to 4.

C. If continuations, return to 10.

11. If "scan time flag" is off, skips to 11B.

A. If time field data is present, converts time to binary for scan storage.

B. If no inputs have been encountered, prints out message with step and substep, and skips to 4.

C. If "new case in progress" flag is on, processes "print storage" inputs in DNS format.

D. If "SEMI flag" is off, skips to 11F.

E. If "SEMI flag" is on, processes "print SEMI" buffer in DNS format, and zero's "SEMI flag".

F. If DISO "time flag" is off, skips to 11H.

G. If DISO "time flag" is on, updates DNS clock time, transfers DISO names and values from temporary storage to print storage with updated times, zero's "time flag" and returns to 11C.

H. If "scan time" flag is off, ends case and skips to 4.

I. If "scan time" flag is on, updates DNS clock time by adding scan duration time, ends case, and returns to 4.

12. If ATOLL operator was a SEMI instruction, checks to see if this is the first new input encountered. If "new start" flag is on, skips to 12B.

A. If "new start" is off, sets "new start" flag on and processes a new DNS *STEP record for this step-substep number, processes the variables field in DNS input format, increments DNS clock input time, and returns to 4.

B. If "DISO input" flag is on, skips to 12D.

C. If "DISO input" flag is off, and no predictions have been encountered, process variables field of record into DNS format, and return to 4.

D. If "PRINT SEMI" buffer is already full, skips to 4.

E. If "PRINT SEMI" buffer is empty, store variables field in print SEMI, and return to 4.

13. If ATOLL operator is a test instruction, checks to see if "new start" flag is on. If "new start" flag is on, skips to 13B.

A. If "new start" is off, sets "new start" flag on, and processes a new DNS *STEP record for this step-substep number before proceeding.

B. Writes out the branching data in the variables field, and returns to 4.

OUTPUT FORMAT:

Figures 3-1 and 3-2 illustrate the type of printout generated by the ATOLL to DNS Conversion and Punch Program. The numbers of the IDA's for IBM tape and TP's for Boeing tapes will be listed as shown in Figure 3-1. Only one IDA is listed in this example but if additional IDA's were being processed they would be listed at this point also. For each IDA list a separate printout will be made similar to the example shown in Figure 3-1.

The word *NAME and IDA 1003 is listed as extracted from the IBM tape for reference and is preceeded by a single asterisk. The test procedure step number is listed following a single asterisk. Cards for these single asterisk comments are punched but are handled by the DNS Simulation Program as comments and are to be used for reference only.

Three asterisks indicate a manual check of the test is to be made to ascertain the mode of simulation corresponds to mode of test and that appropriate inputs are added as required. The actual inputs are listed without any preceeding asterisks as shown, DO34 = 1 at 50.

Punched cards that require removal from the input card deck are "Input Requirement" type comment cards. Inset 7 card columns the comment "INPUT REQ-T" (requirement) will appear followed by a comment. The sixth line of the printout is shown in Figure 3-1 which calls the users attention to the fact that at this point in the test procedure, an input not automatically generated, is required to start power supply 6D100. Investigation reveals that in this case it is a manual push button. Therefore the comment card is removed, and a new card inputting the required push button will be inserted in its place.

A sequence number is assigned to each card that is punched as shown in Figure 3-1. In event that any such inputs require more than one card to replace the comment card, it would be advisable to insert all changes to the Update Program to ensure a new sequence number. Present DNS/simulation requirements do not merit a change from punched card driving inputs. The input conversion and punch program was written with future applications in mind where tape inputs would be a necessity or beneficial.

```
        IBM TEST PRØCEDURE NUMBERS

                IDA5003


GENERAL DYNAMICS/CØNVAIR ATØLL TAPE TØ DNS INPUT CØNVERSIØN AND PUNCH PRØGRAM  PAGE NØ   1

 ATØLL TEST PRØCEDURE NØ IDA5003

 *     *NAME      IDA5003                                                  SEQ    1

 *     *STEP NØ    002000                                                  SEQ    2

 ***  TEST WITH BRANCH TØ STEP 002008 ENCØUNTERED- CHECK FØR EXTRA INPUTS

 D034      = 1 AT      50.                                                 SEQ    3

 *     *STEP NØ    003000                                                  SEQ    4

           INPUT REQ-T        PRESS 6D100 START BUTTØN                     SEQ    5

 *     *STEP NØ    004000                                                  SEQ    6

           INPUT REQ-T        6D100 ØUTPUT PØWER SWITCH ØN                 SEQ    7

 *     *STEP NØ    005000                                                  SEQ    8

           INPUT REQ-T        6D100BAT ENABLE SW ØN                        SEQ    9

 *     *STEP NØ    006000                                                  SEQ   10

           INPUT REQ-T        TURN ØUTPUT PWR SW ØFF                       SEQ   11
```

Figure 3-1. Punch card and manual input listing, input conversion program.

A summary of each IDA is listed. Each "CASE" is from step number to the last DO input prior to the next step number. The number of cards punched are totaled. The *END data card is required to make program compatible with the Update Program and will only appear after the final IDA, if more than one is processed. Figure 3-2 represents the final portion of the test procedure generated inputs. The word *END signals the termination of the data for the DNS simulation.

```
GENERAL DYNAMICS/CØNVAIR ATØLL TAPE TØ DNS INPUT CØNVERSIØN AND PUNCH PRØGRAM  PAGE NØ  41

ATØLL TEST PRØCEDURE NØ IDA5003

*     *STEP NØ    055000                                                  SEQ 964

D088      = 1 AT   28550.                                                 SEQ 965

*     *STEP NØ    055200                                                  SEQ 966

D087      = 1 AT   28600.                                                 SEQ 967

*     *STEP NØ    055500                                                  SEQ 968

***  TEST WITH BRANCH TØ STEP 055700 ENCØUNTERED- CHECK FØR EXTRA INPUTS

D0274     = 1 AT   28650.                                                 SEQ 969

D0274     = 0 AT   28700.                                                 SEQ 970

D0275     = 1 AT   28700.                                                 SEQ 971

D0275     = 0 AT   28750.                                                 SEQ 972

            INPUT REQ-T       VERIFY PSU IS NØT VENTING$                  SEQ 973

*     *END                                                                SEQ 974


SUMMARY - TEST PRØCEDURE NUMBER IDA5003
      NUMBER ØF CASES =  235
      NUMBER ØF CARDS =  974

*END DATA      DNS/ATØLL INPUT CØNVERSIØN AND PUNCH PRØGRAM GENERATED INPUTS
```

Figure 3-2. Punch card and data total, input conversion program.

APPENDIX A

PROGRAM FLOW CHARTS

PRECEDING PAGE BLANK NOT FILMED

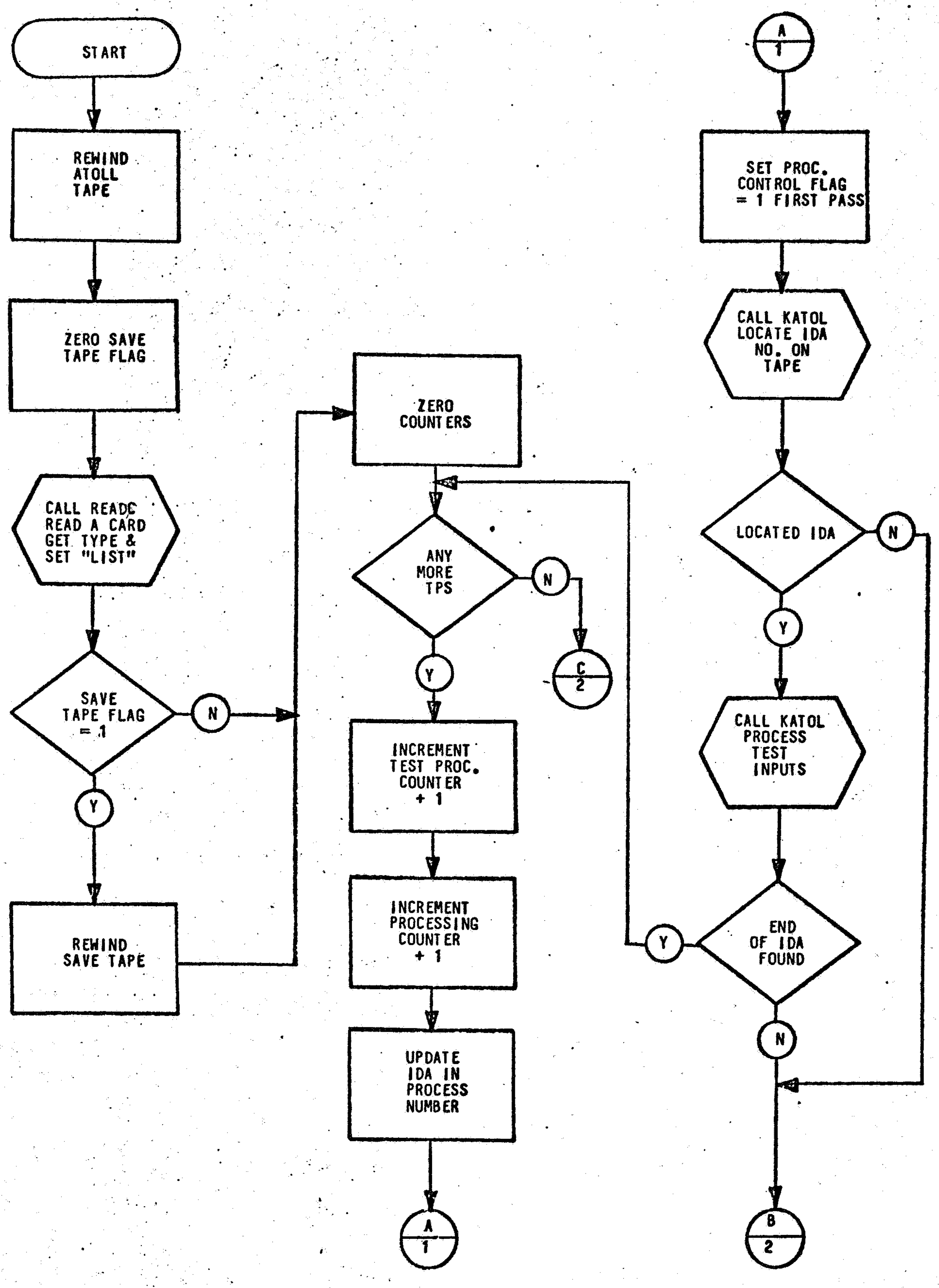

START
REWIND ATOLL TAPE
ZERO SAVE TAPE FLAG
CALL READC READ A CARD GET TYPE & SET "LIST"
SAVE TAPE FLAG = 1
N
Y
REWIND SAVE TAPE
ZERO COUNTERS
ANY MORE TPS
N
C 2
Y
INCREMENT TEST PROC. COUNTER + 1
INCREMENT PROCESSING COUNTER + 1
UPDATE IDA IN PROCESS NUMBER
A 1
A 1
SET PROC. CONTROL FLAG = 1 FIRST PASS
CALL KATOL LOCATE IDA NO. ON TAPE
LOCATED IDA
N
Y
CALL KATOL PROCESS TEST INPUTS
END OF IDA FOUND
Y
N
B 2

Figure A-1. SUBROUTINE CONTRD (1 of 2)

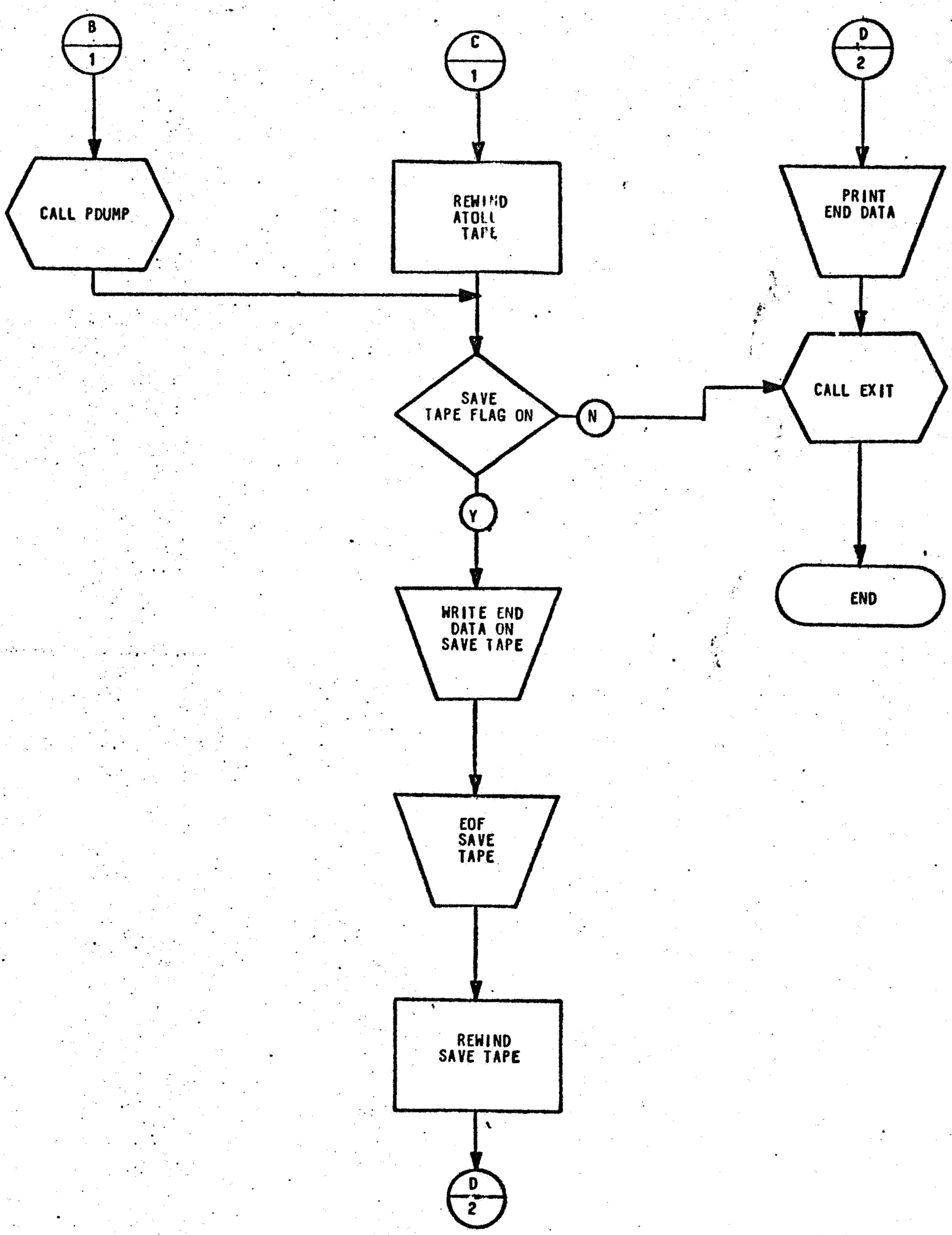


Figure A-1. SUBROUTINE CONTRD (2 of 2)

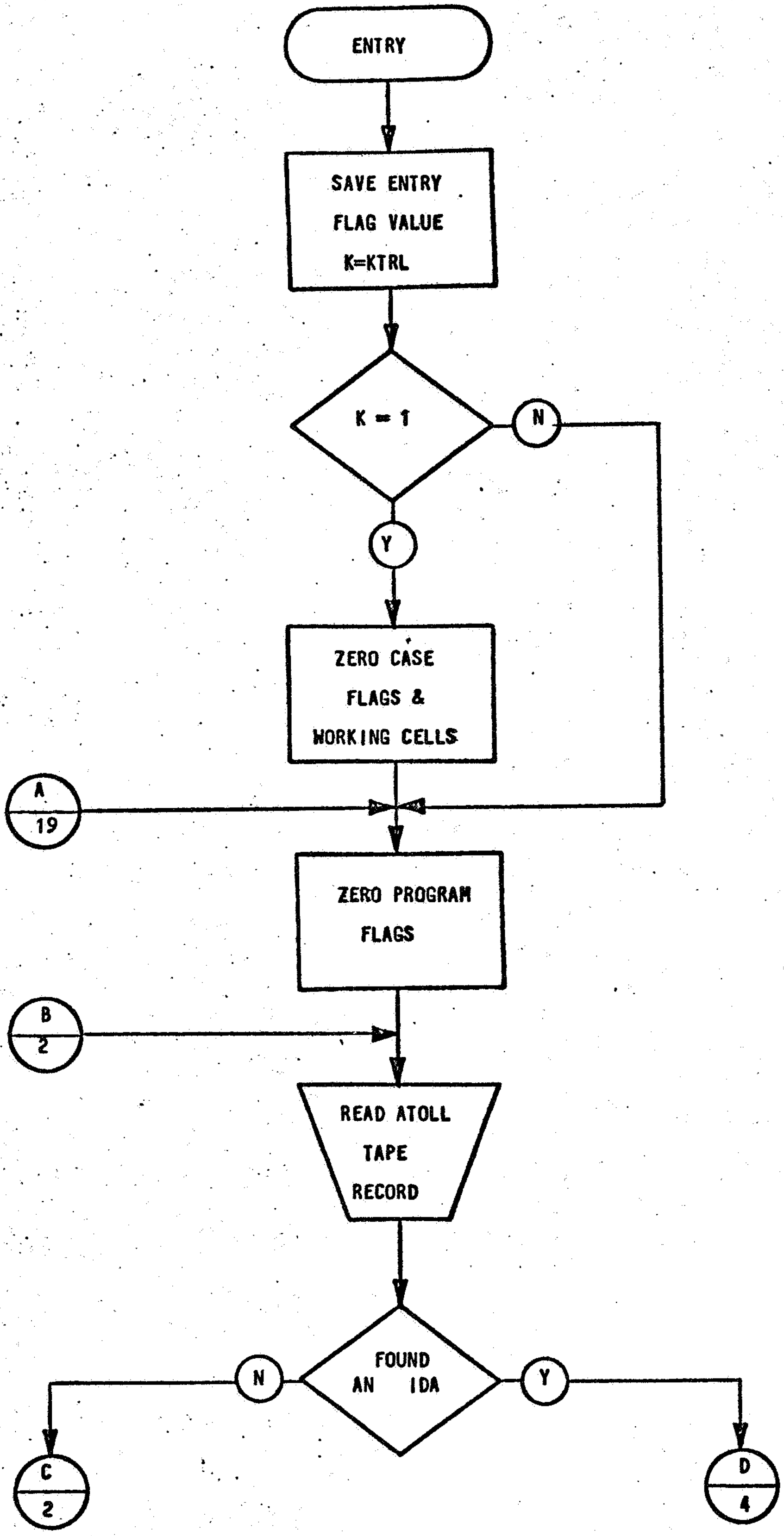


Figure A-2. SUBROUTINE KATLT (1 of 23)

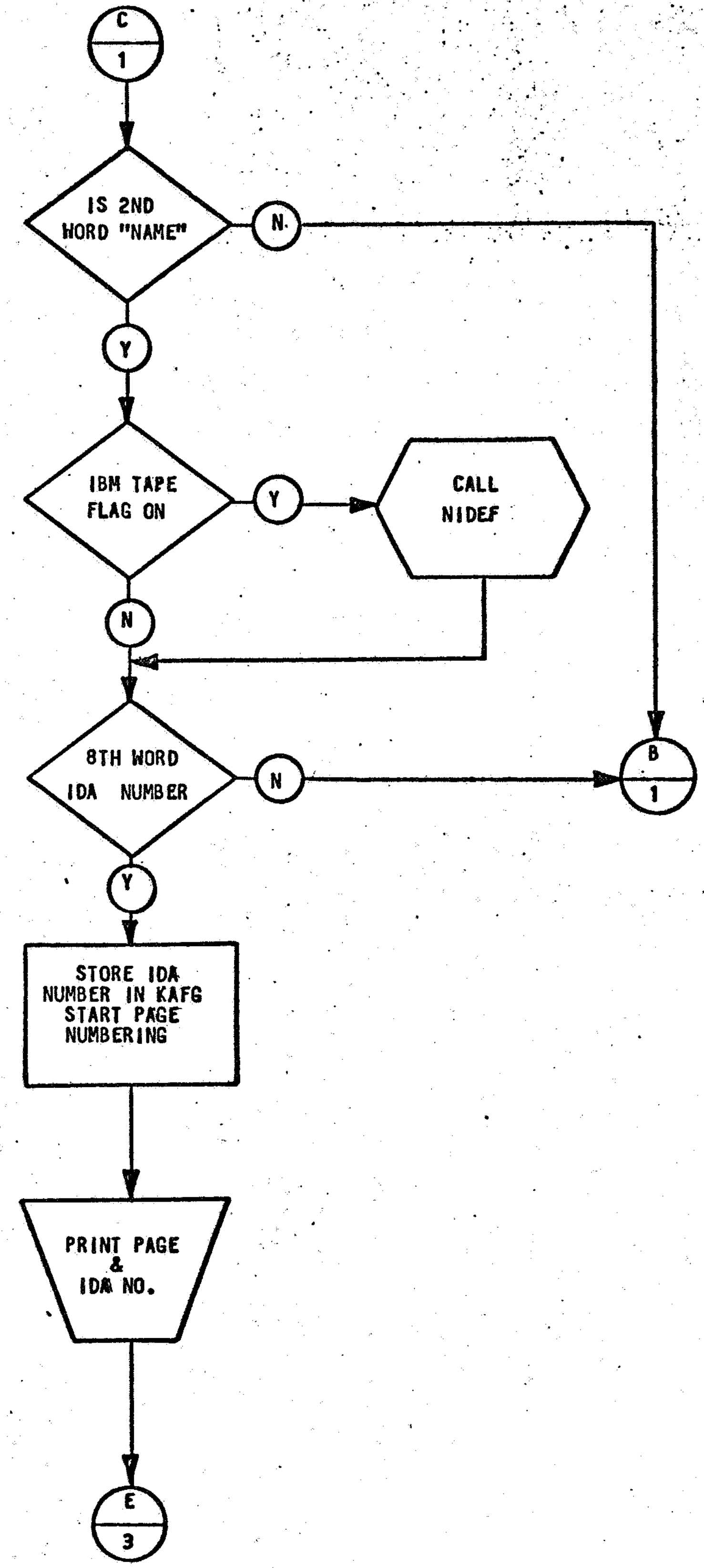


Figure A-2. SUBROUTINE KATLT (2 of 23)

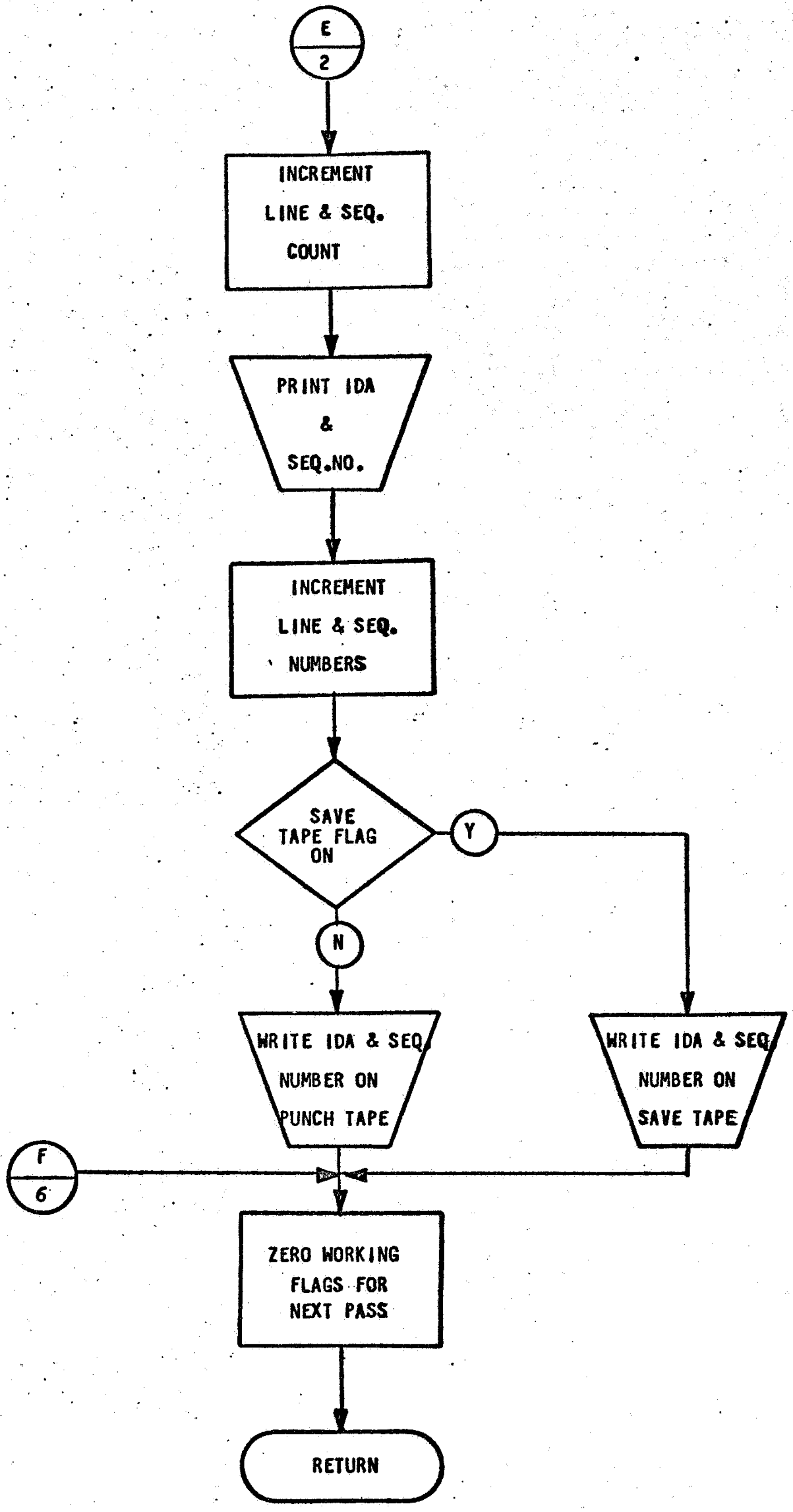

E
2
INCREMENT LINE & SEQ. COUNT
PRINT IDA & SEQ.NO.
INCREMENT LINE & SEQ. NUMBERS
SAVE TAPE FLAG ON
Y
N
WRITE IDA & SEQ. NUMBER ON PUNCH TAPE
WRITE IDA & SEQ. NUMBER ON SAVE TAPE
F
6
ZERO WORKING FLAGS FOR NEXT PASS
RETURN

Figure A-2. SUBROUTINE KATLT (3 of 23)

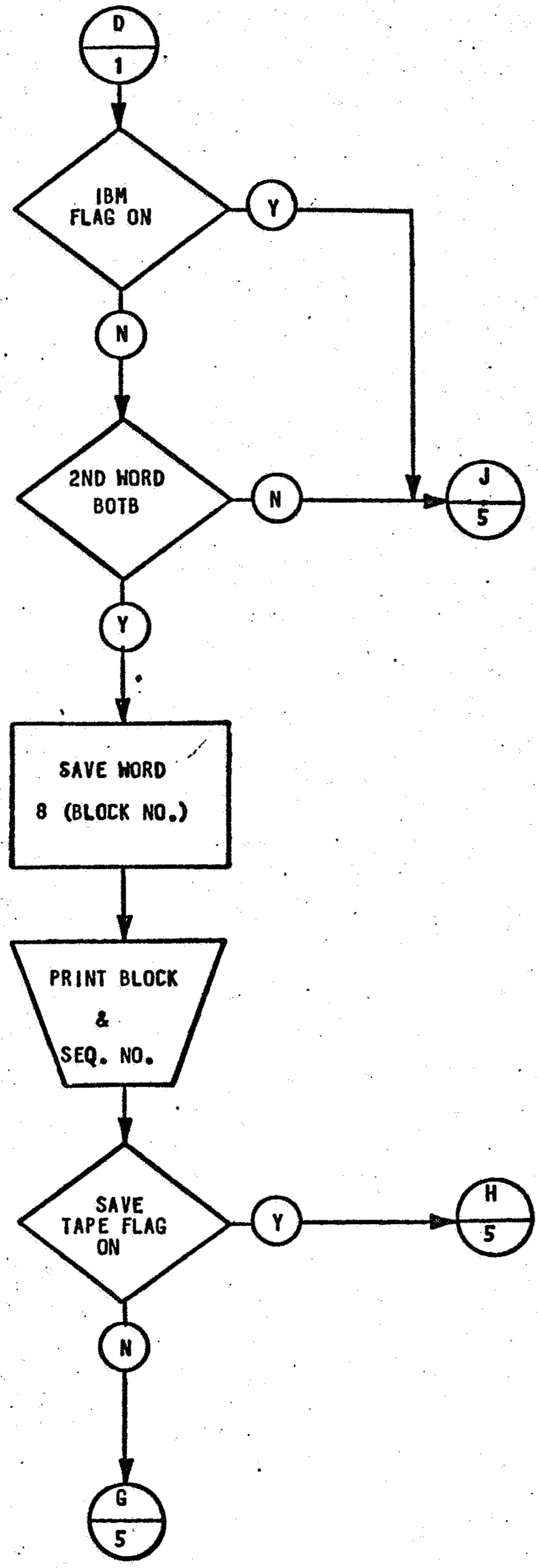


Figure A-2. SUBROUTINE KATLT (4 of 23)

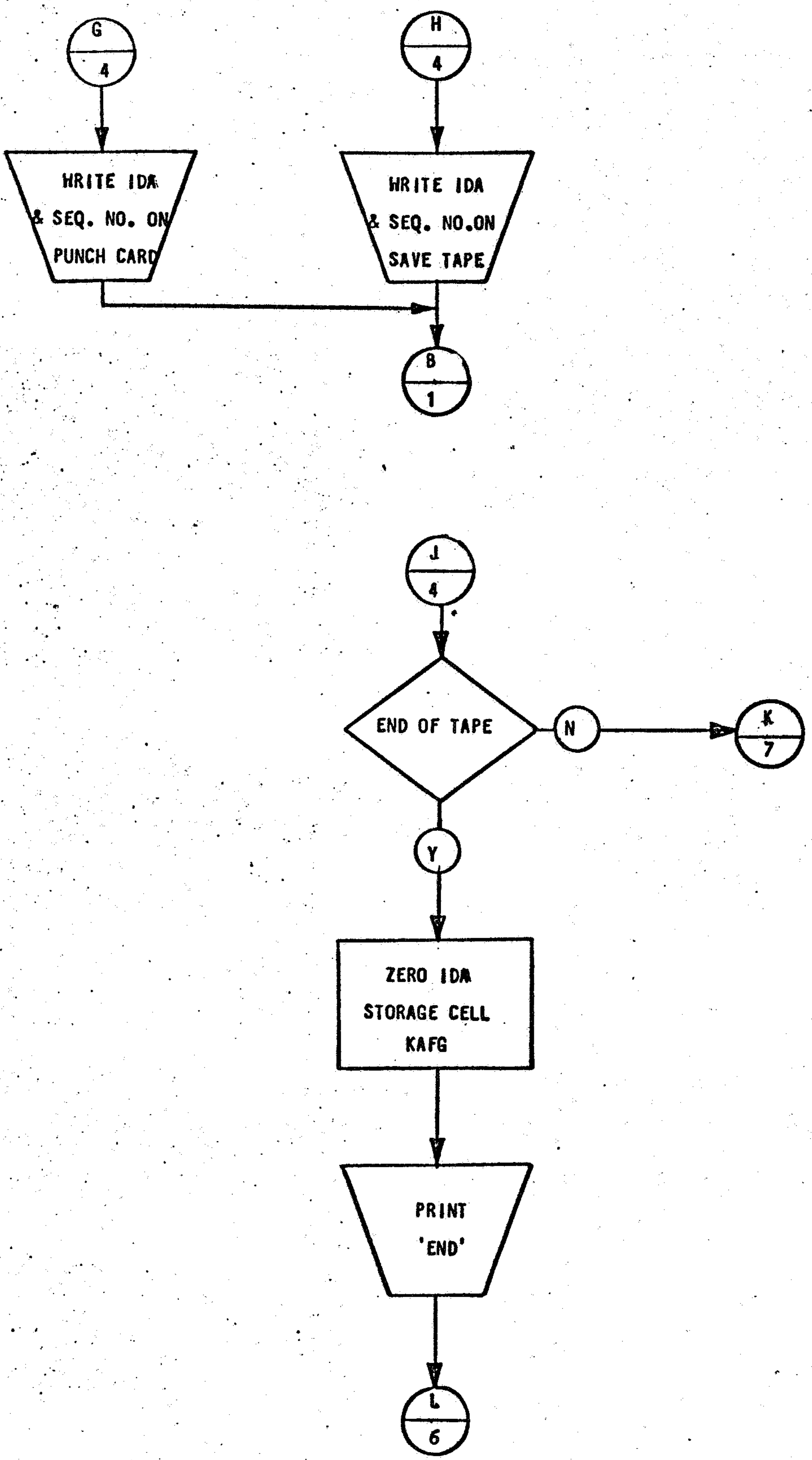

G
4
WRITE IDA & SEQ. NO. ON PUNCH CARD
H
4
WRITE IDA & SEQ. NO. ON SAVE TAPE
B
1
J
4
END OF TAPE
N
K
7
Y
ZERO IDA STORAGE CELL KAFG
PRINT 'END'
L
6

Figure A-2. SUBROUTINE KATLT (5 of 23)

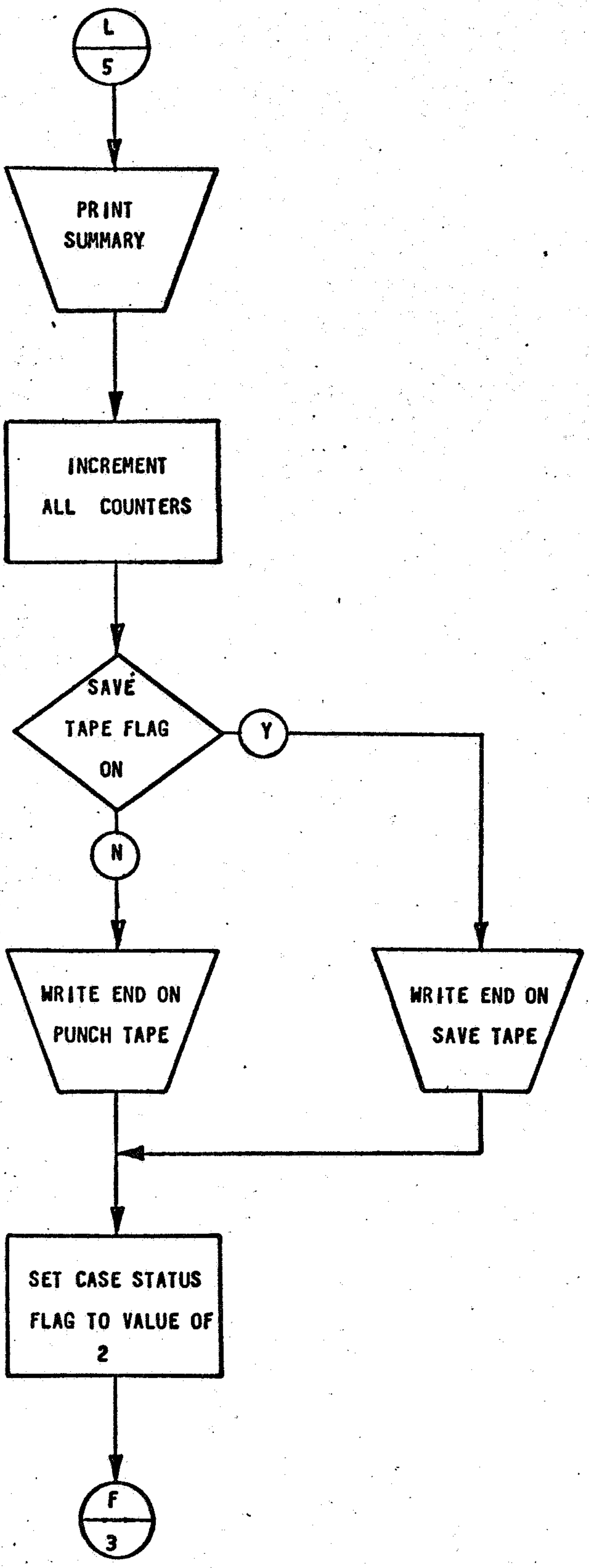


Figure A-2. SUBROUTINE KATLT (6 of 23)

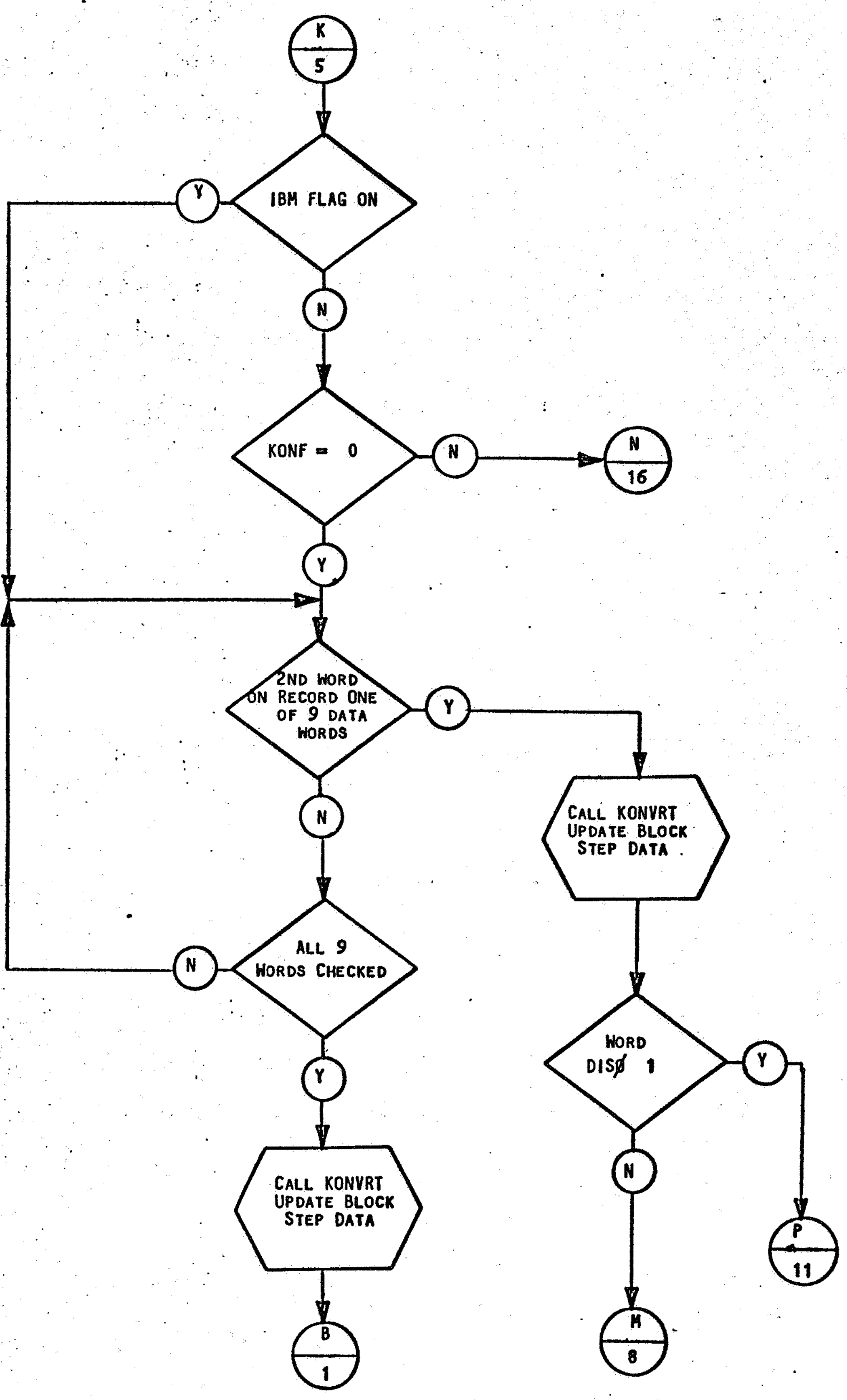

K
5
IBM FLAG ON
Y
N
KONF = 0
N
N
16
Y
2ND WORD ON RECORD ONE OF 9 DATA WORDS
Y
N
CALL KONVRT UPDATE BLOCK STEP DATA
ALL 9 WORDS CHECKED
N
WORD DISØ 1
Y
Y
N
CALL KONVRT UPDATE BLOCK STEP DATA
P
11
B
1
M
8

Figure A-2. SUBROUTINE KATLT (7 of 23)

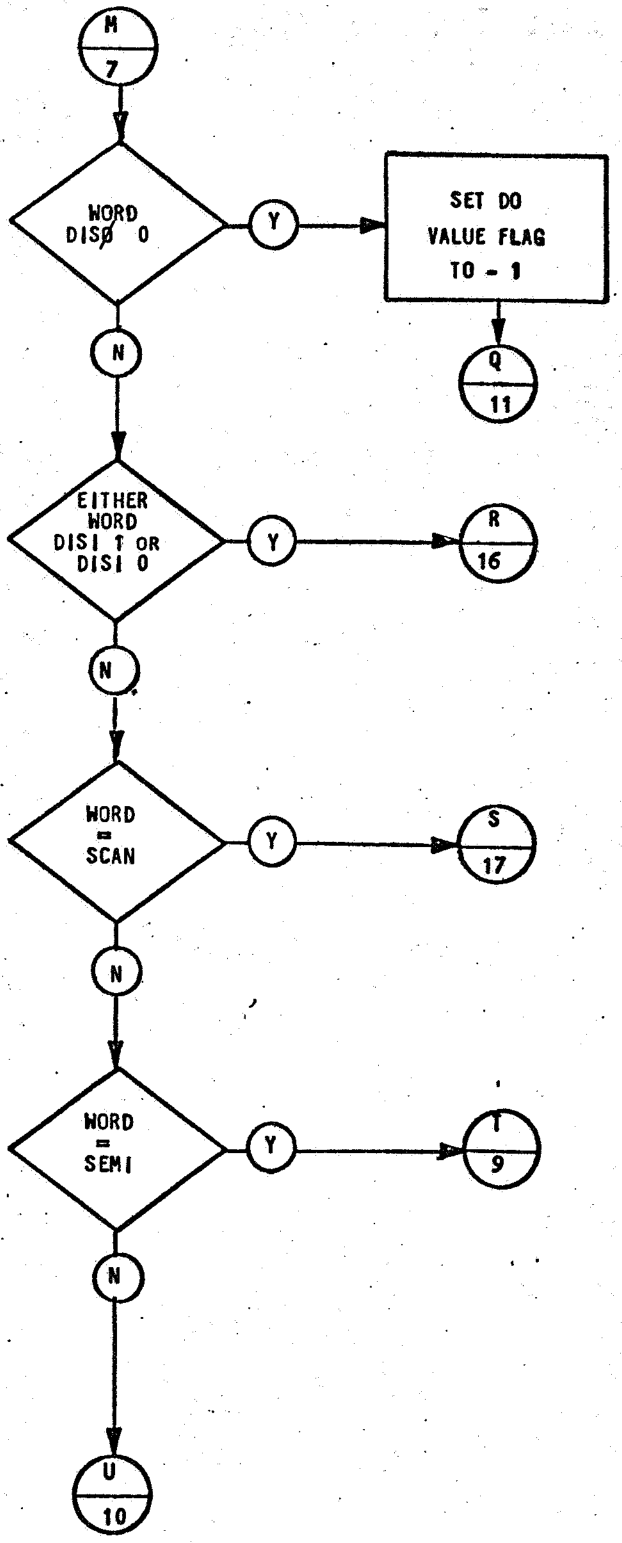


Figure A-2. SUBROUTINE KATLT (8 of 23)

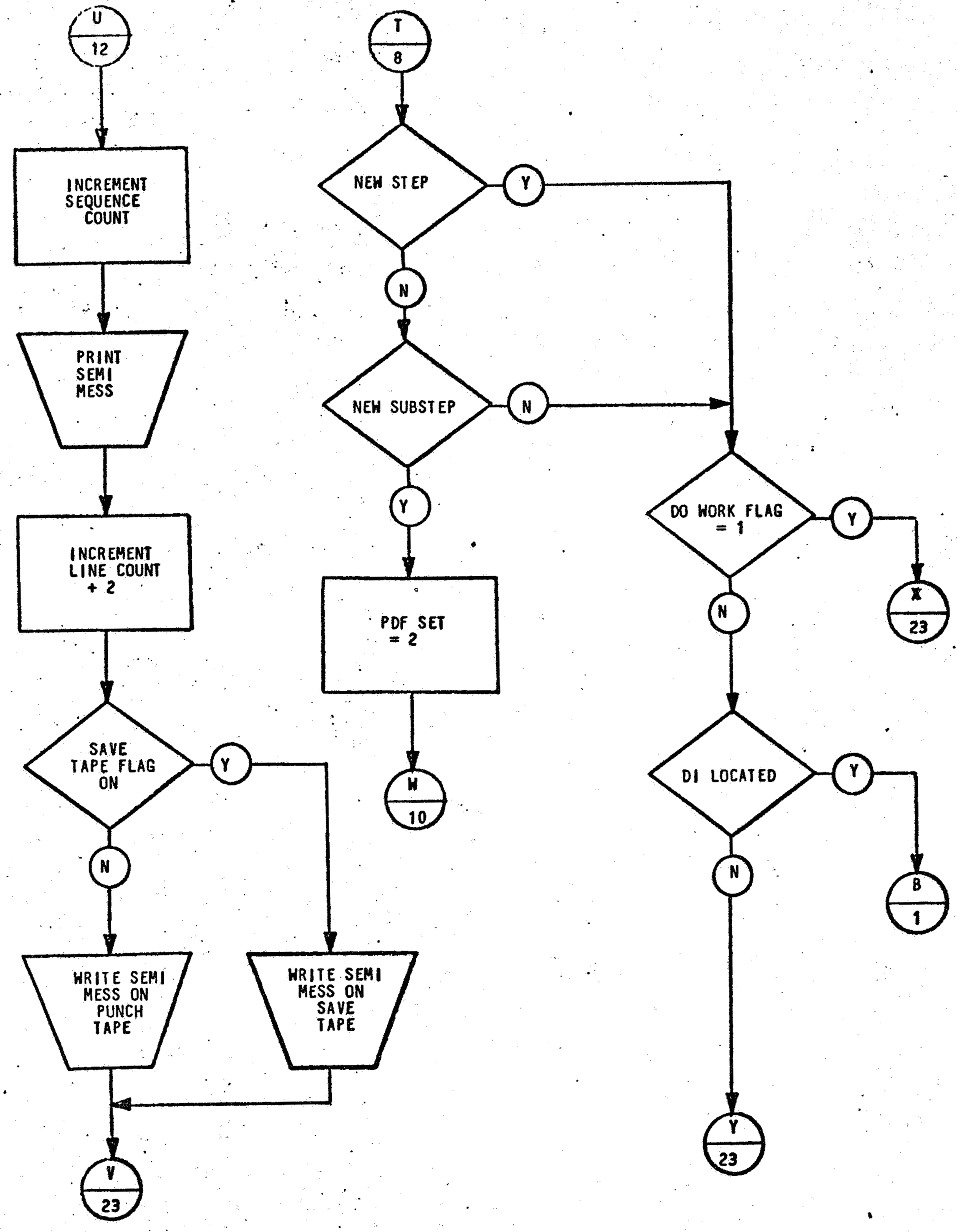


Figure A-2. SUBROUTINE KATLT (9 of 23)

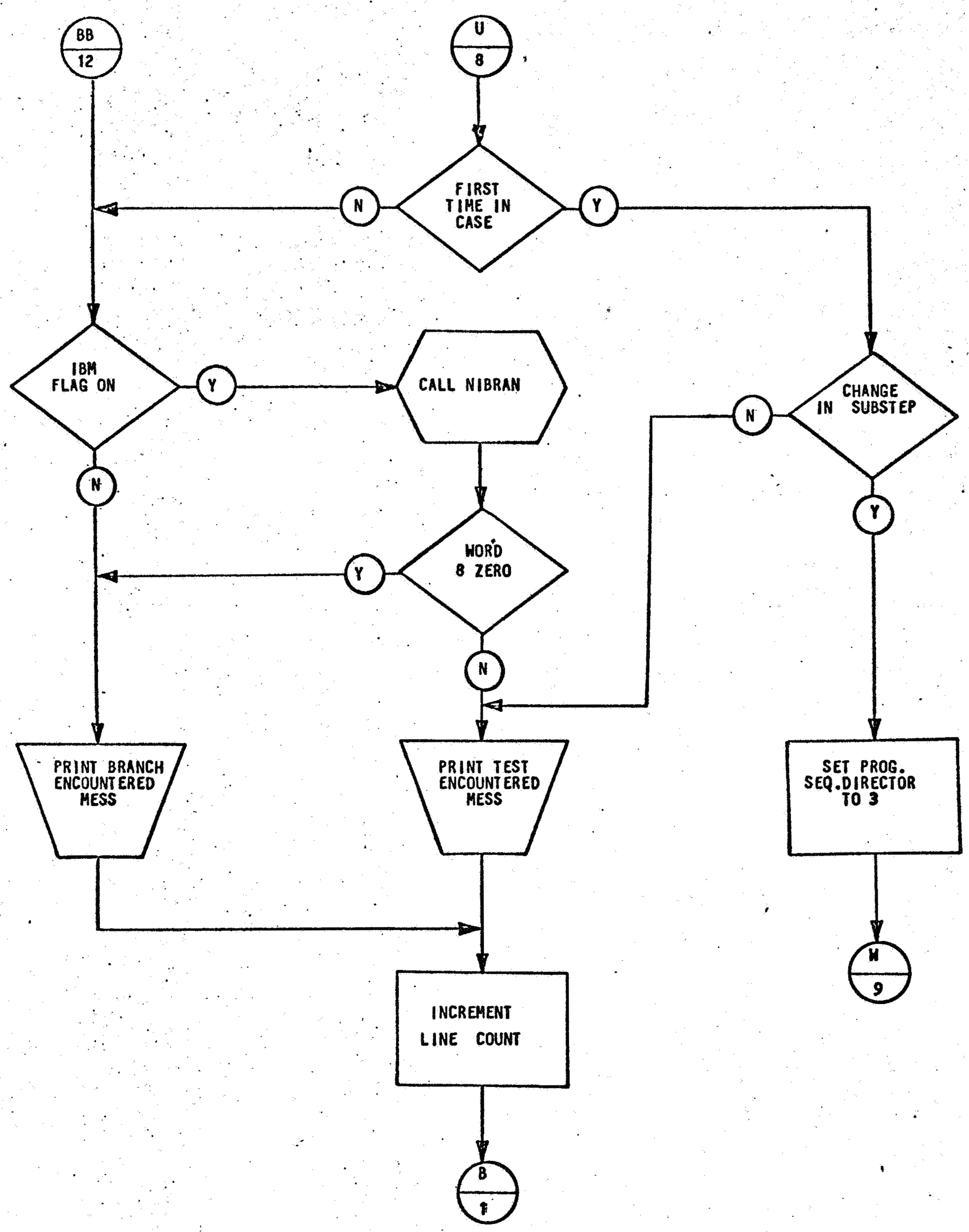


Figure A-2. SUBROUTINE KATLT (10 of 23)

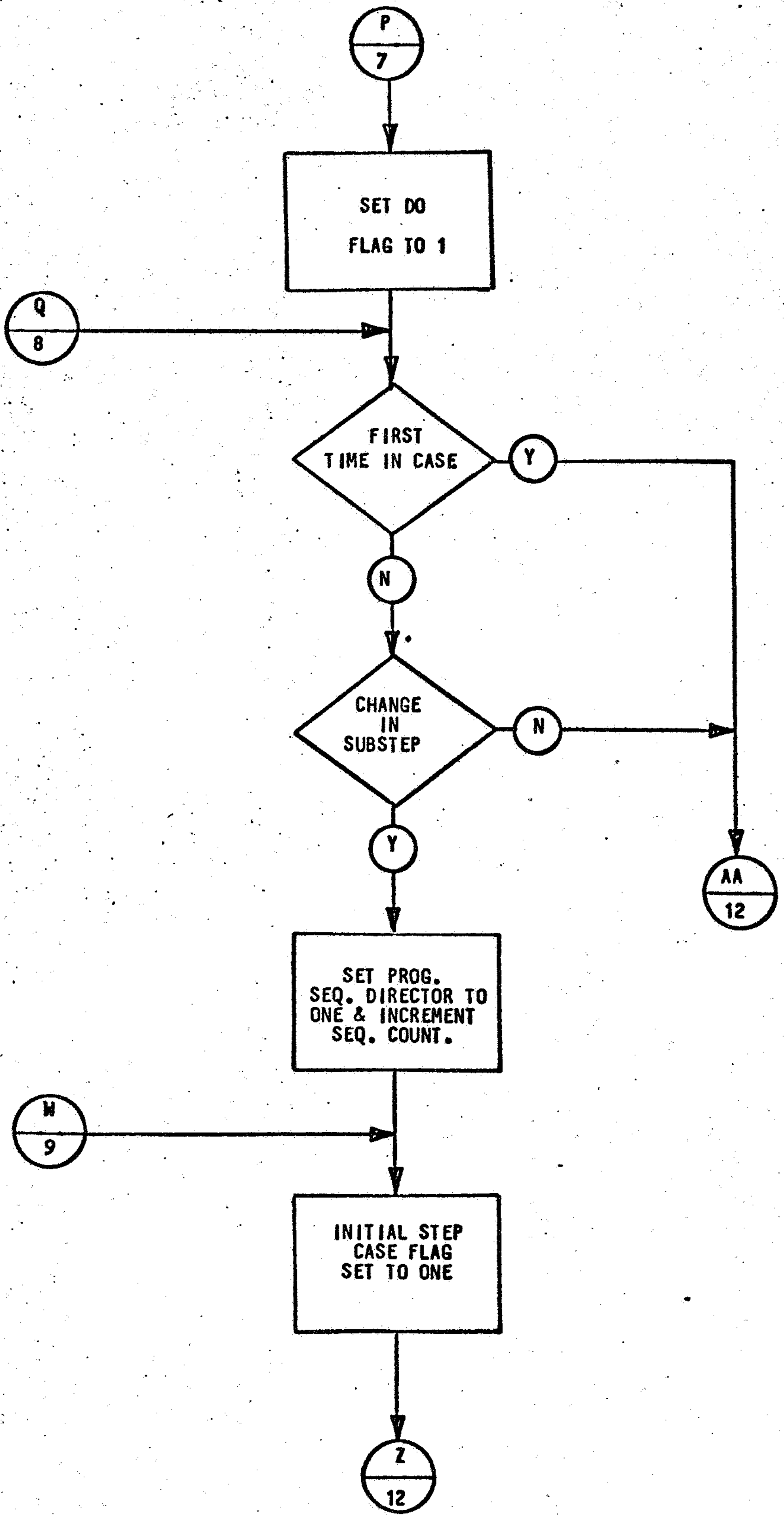

P
7
SET DO FLAG TO 1
Q
8
FIRST TIME IN CASE
Y
N
CHANGE IN SUBSTEP
N
Y
AA
12
SET PROG. SEQ. DIRECTOR TO ONE & INCREMENT SEQ. COUNT.
W
9
INITIAL STEP CASE FLAG SET TO ONE
Z
12

Figure A-2. SUBROUTINE KATLT (11 of 23)

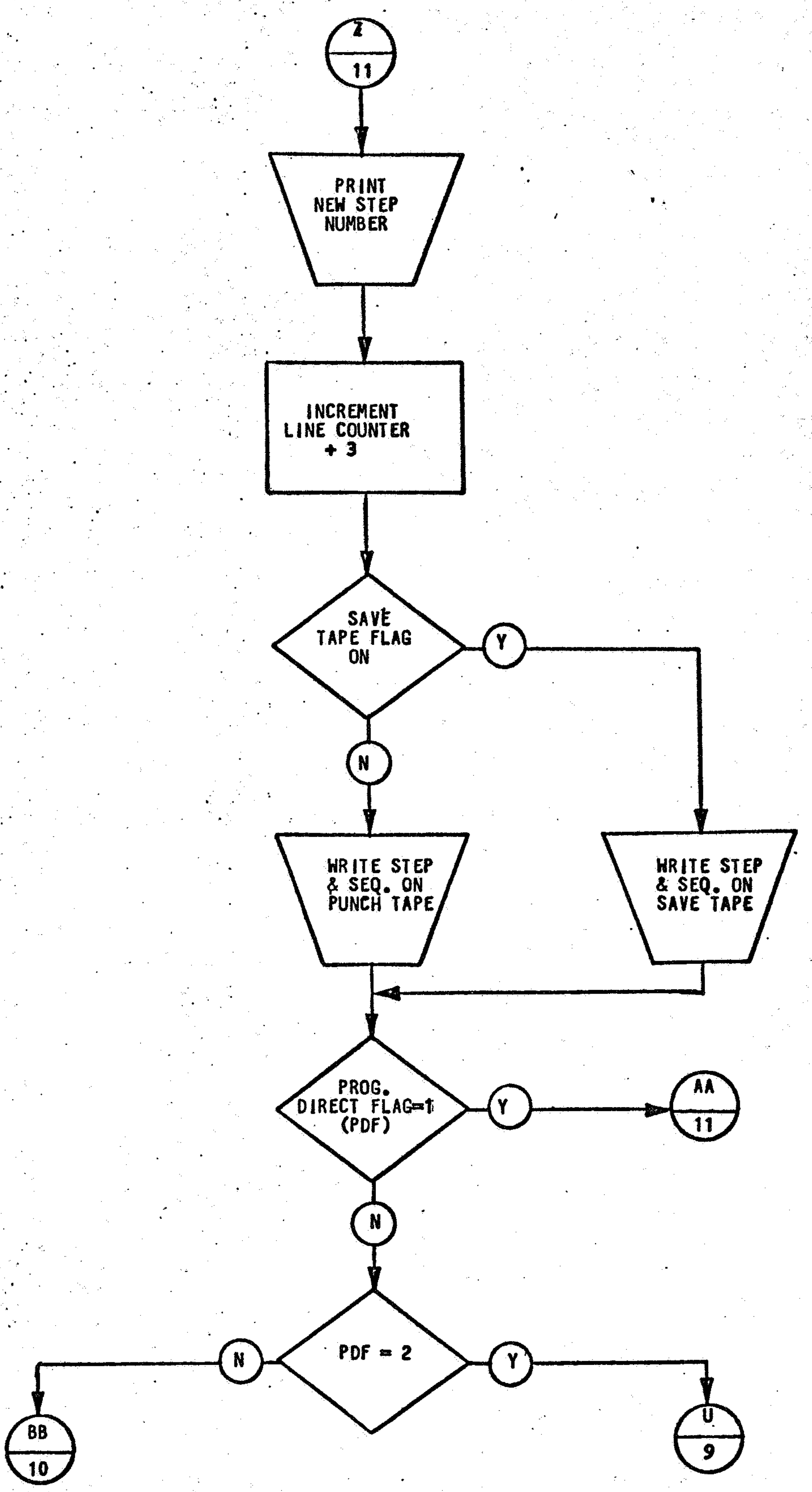

Z
11
PRINT NEW STEP NUMBER
INCREMENT LINE COUNTER + 3
SAVE TAPE FLAG ON
Y
N
WRITE STEP & SEQ. ON PUNCH TAPE
WRITE STEP & SEQ. ON SAVE TAPE
PROG. DIRECT FLAG=1 (PDF)
Y
AA
11
N
PDF = 2
N
Y
BB
10
U
9

Figure A-2. SUBROUTINE KATLT (12 of 23)

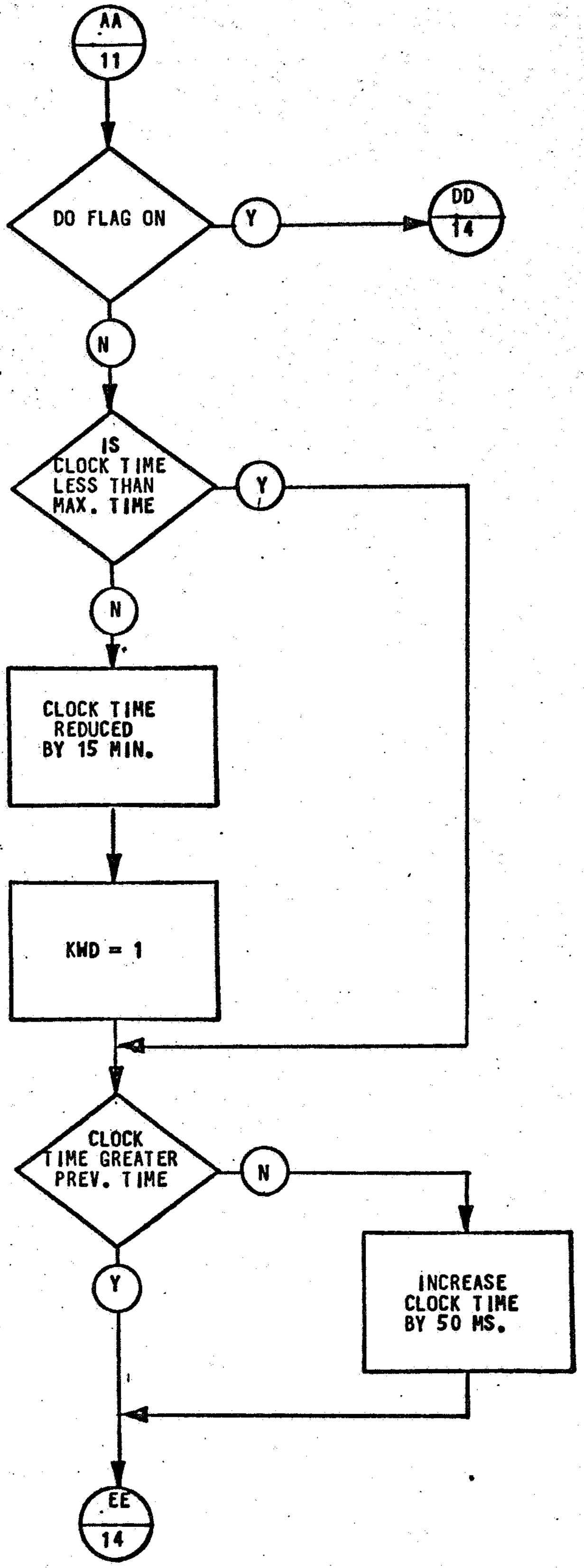

AA
11
DO FLAG ON
Y
DD
14
N
IS CLOCK TIME LESS THAN MAX. TIME
Y
N
CLOCK TIME REDUCED BY 15 MIN.
KWD = 1
CLOCK TIME GREATER PREV. TIME
N
INCREASE CLOCK TIME BY 50 MS.
Y
EE
14

Figure A-2. SUBROUTINE KATLT (13 of 23)

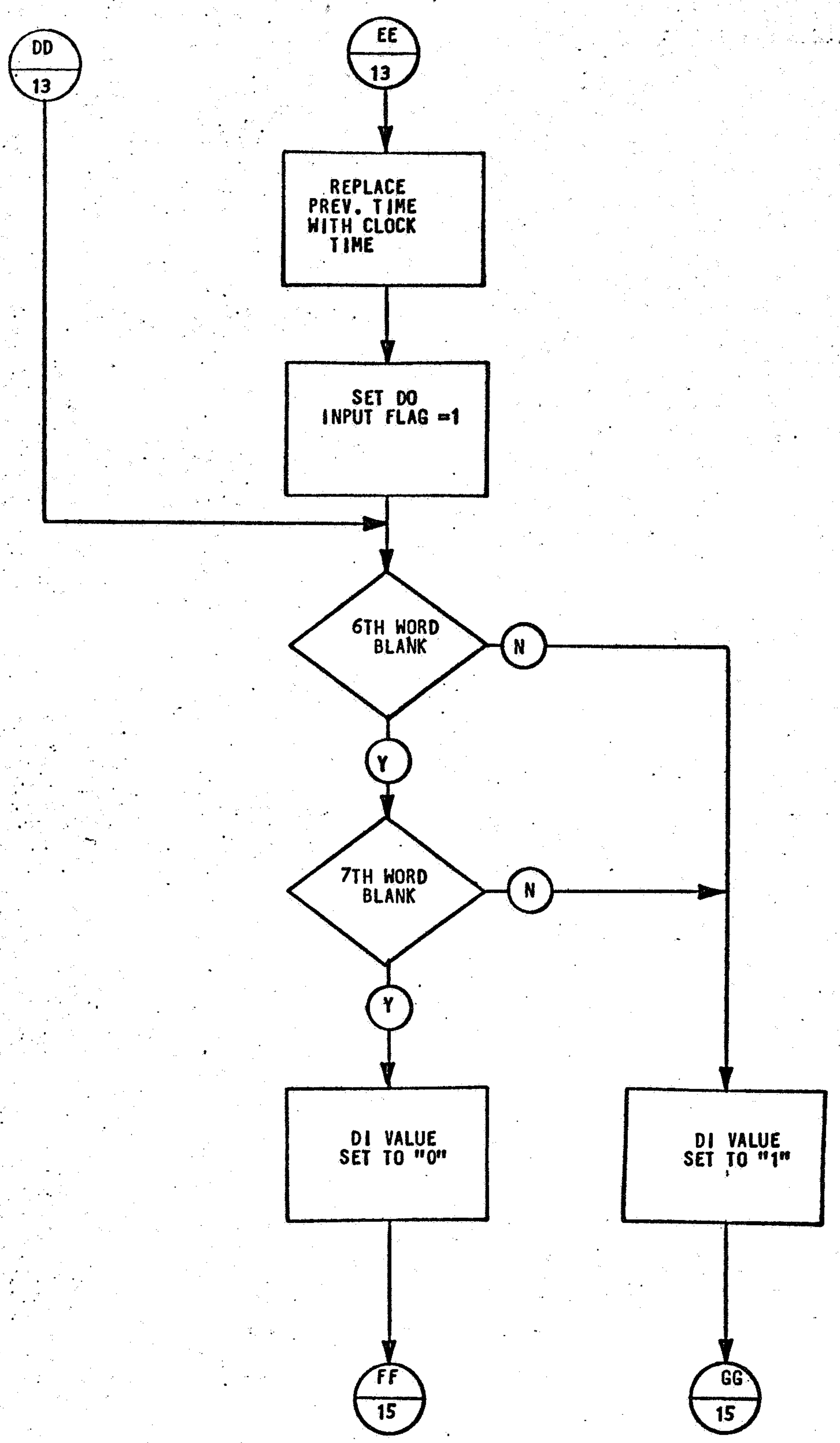


Figure A-2. SUBROUTINE KATLT (14 of 23)

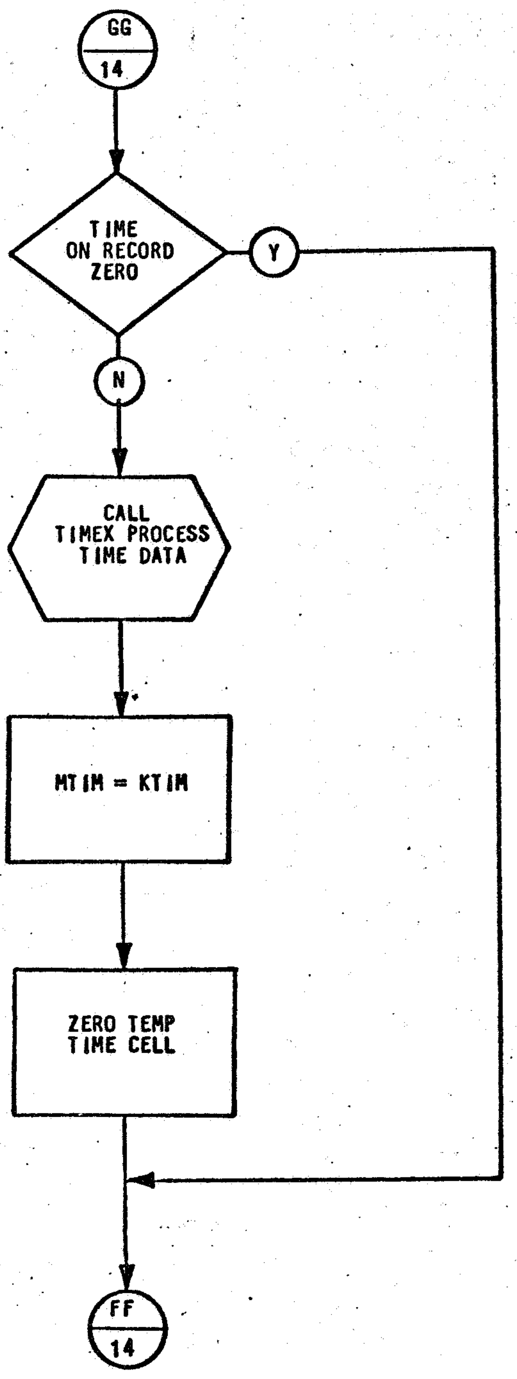


Figure A-2. SUBROUTINE KATLT (15 of 23)

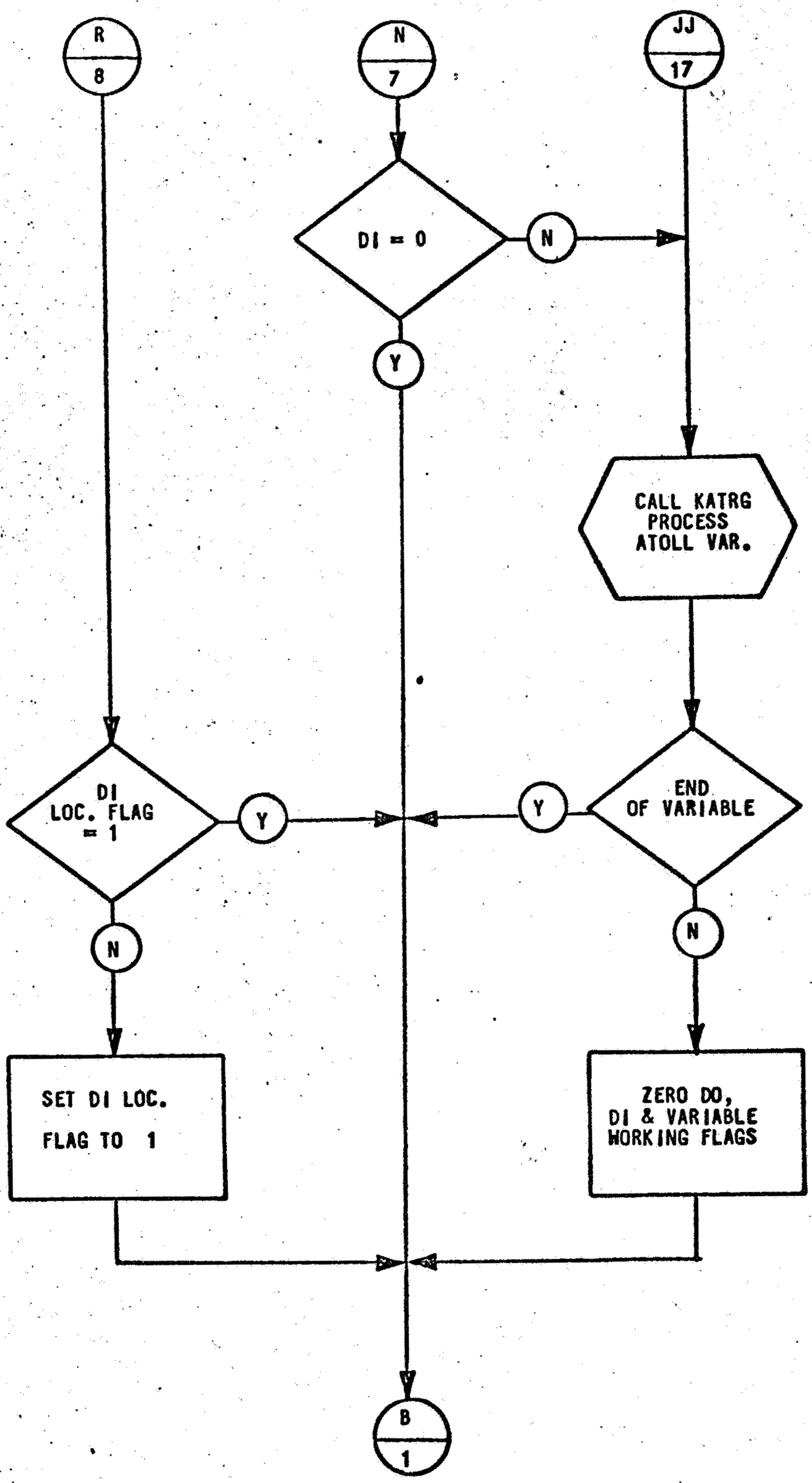


Figure A-2. SUBROUTINE KATLT (16 of 23)

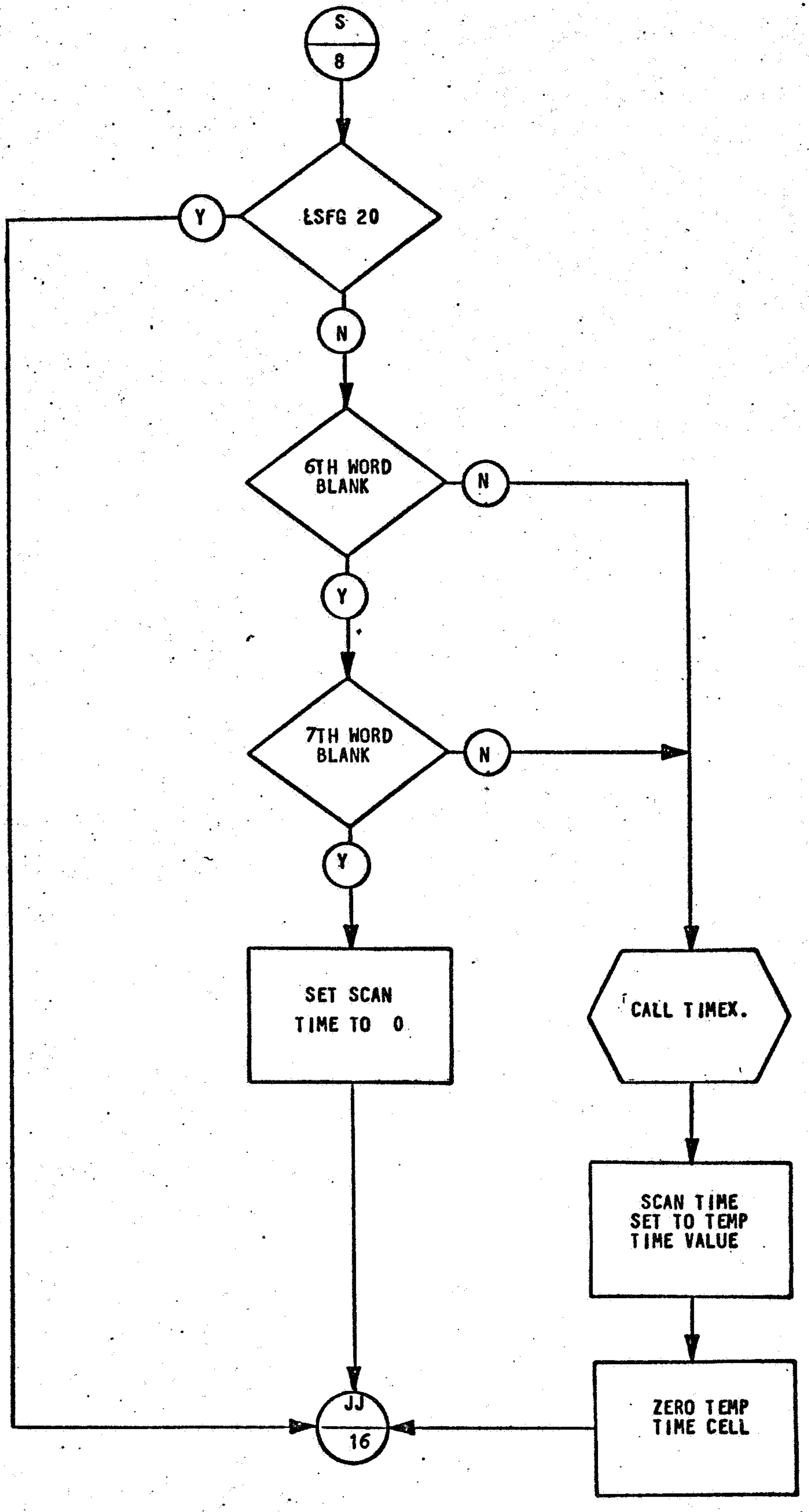


Figure A-2. SUBROUTINE KATLT (17 of 23)

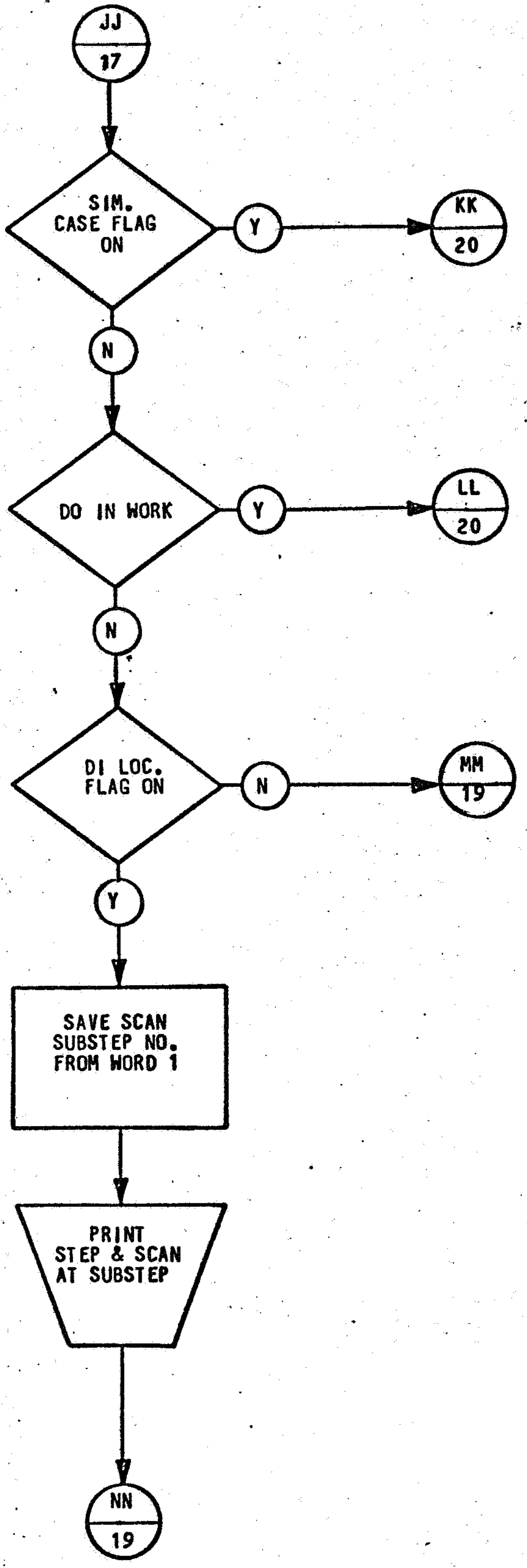


Figure A-2. SUBROUTINE KATLT (18 of 23)

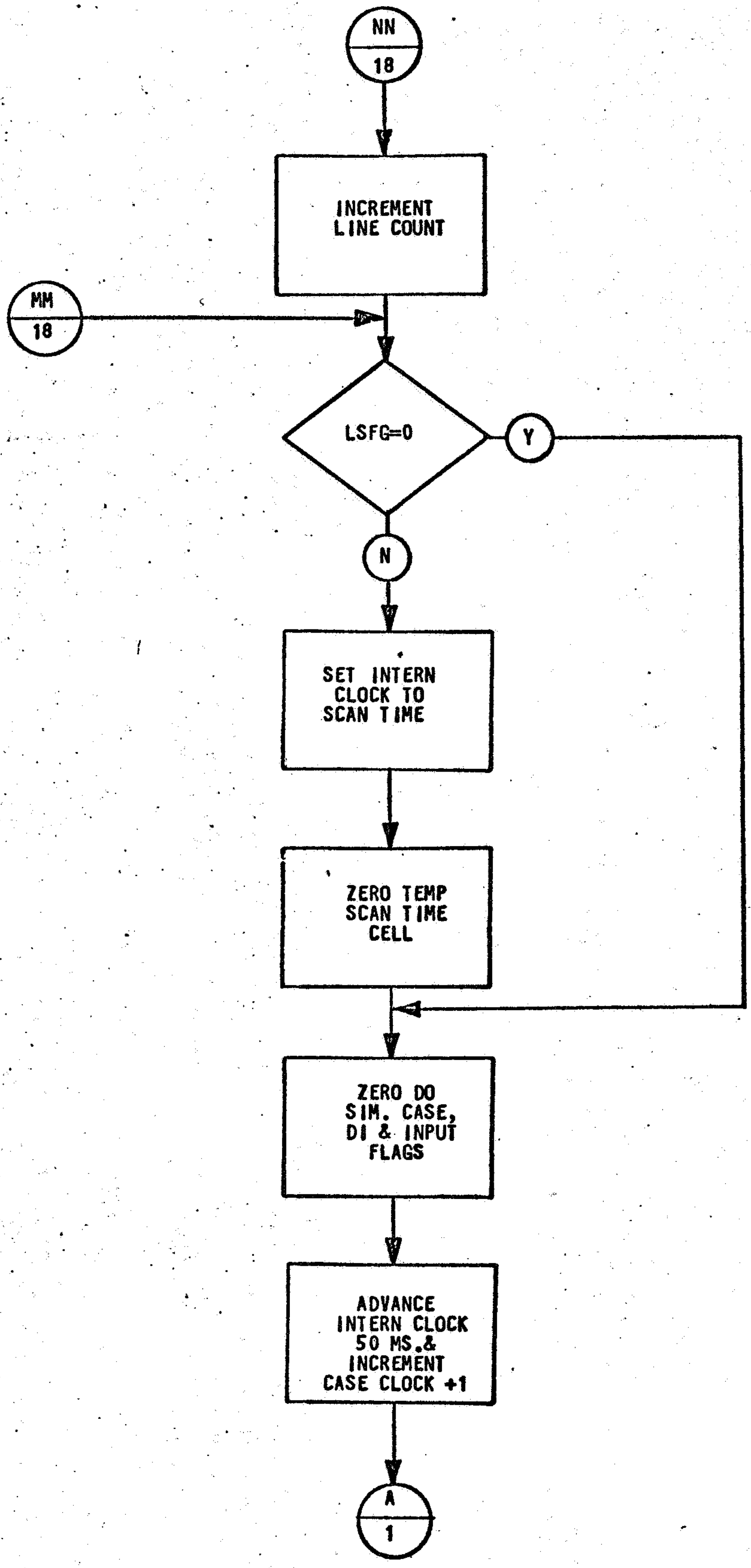

NN
18
INCREMENT LINE COUNT
MM
18
LSFG=0
Y
N
SET INTERN CLOCK TO SCAN TIME
ZERO TEMP SCAN TIME CELL
ZERO DO SIM. CASE, DI & INPUT FLAGS
ADVANCE INTERN CLOCK 50 MS.& INCREMENT CASE CLOCK +1
A
1

Figure A-2. SUBROUTINE KATLT (19 of 23)

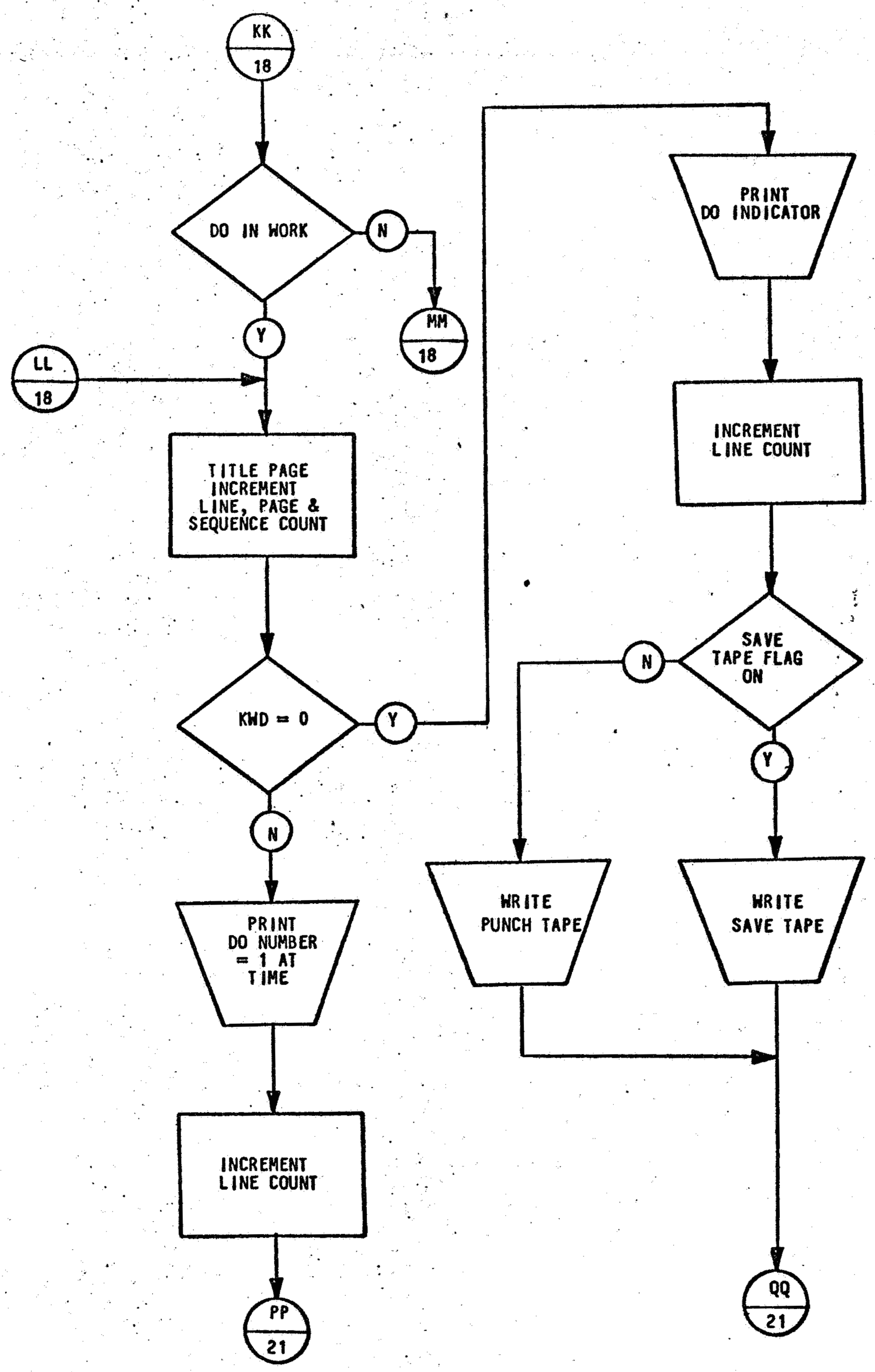


Figure A-2. SUBROUTINE KATLT (20 of 23)

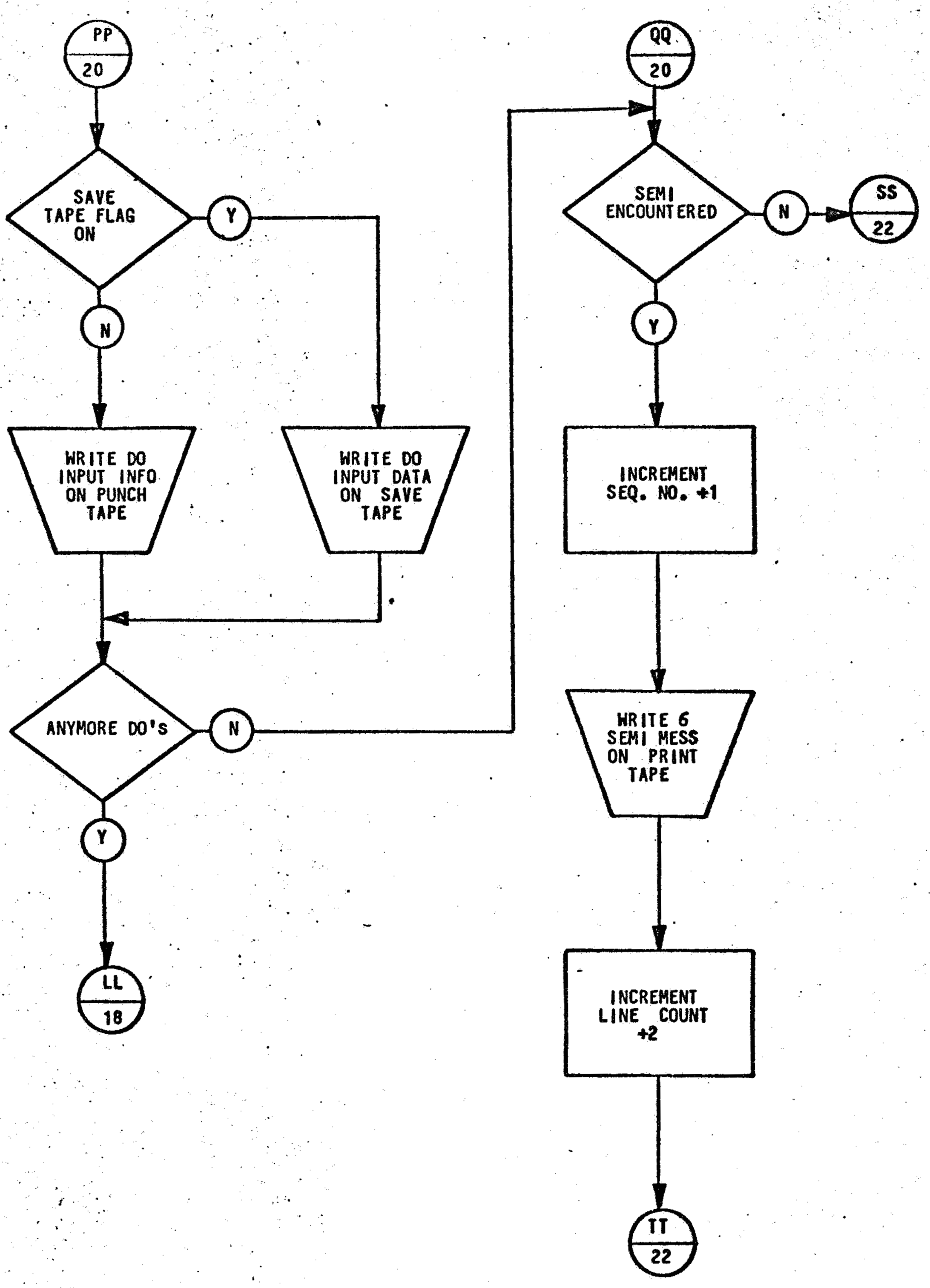


Figure A-2. SUBROUTINE KATLT (21 of 23)

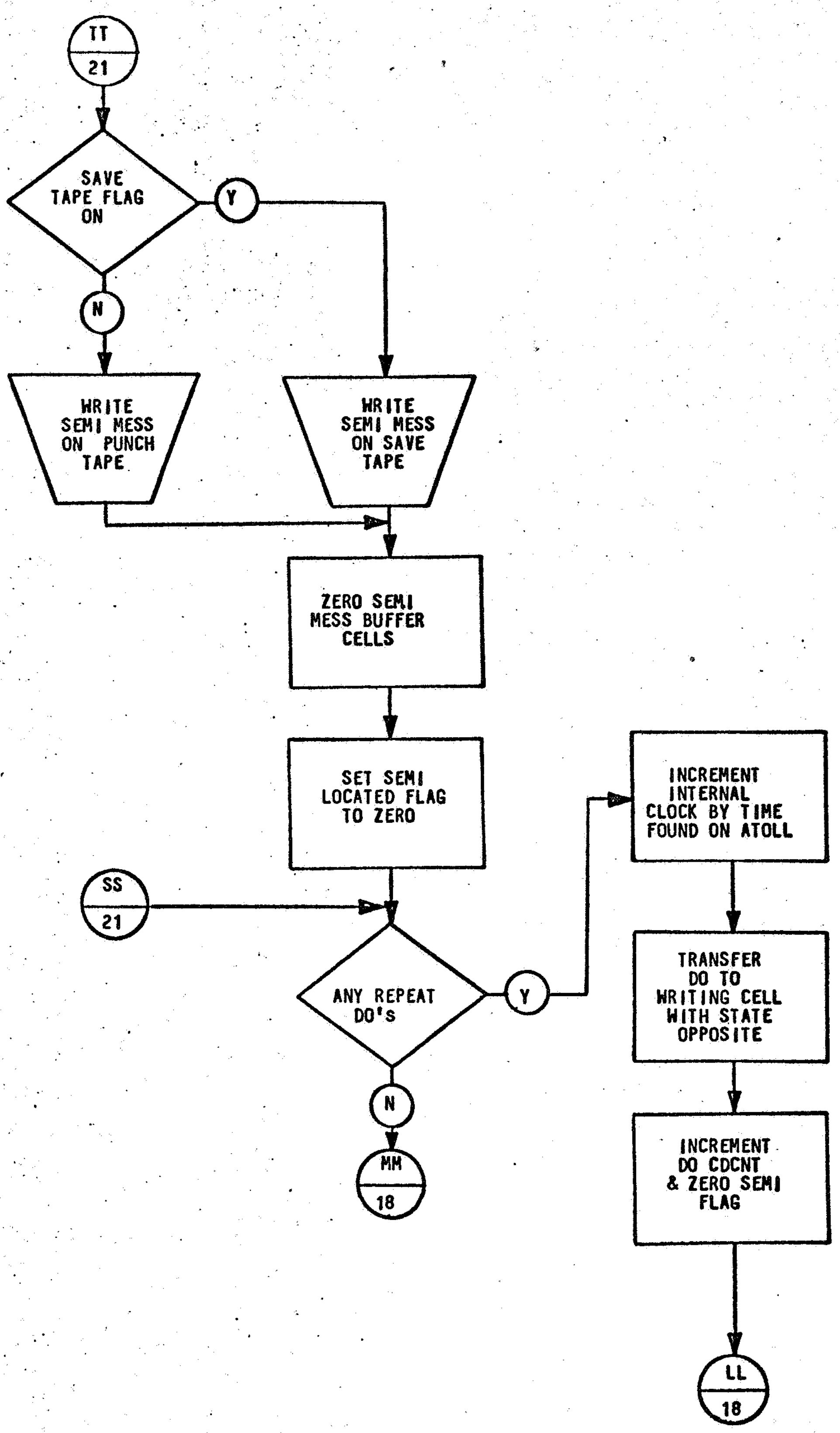

TT
21
SAVE TAPE FLAG ON
Y
N
WRITE SEMI MESS ON PUNCH TAPE
WRITE SEMI MESS ON SAVE TAPE
ZERO SEMI MESS BUFFER CELLS
SET SEMI LOCATED FLAG TO ZERO
SS
21
ANY REPEAT DO's
Y
N
MM
18
INCREMENT INTERNAL CLOCK BY TIME FOUND ON ATOLL
TRANSFER DO TO WRITING CELL WITH STATE OPPOSITE
INCREMENT DO CDCNT & ZERO SEMI FLAG
LL
18

Figure A-2. SUBROUTINE KATLT (22 of 23)

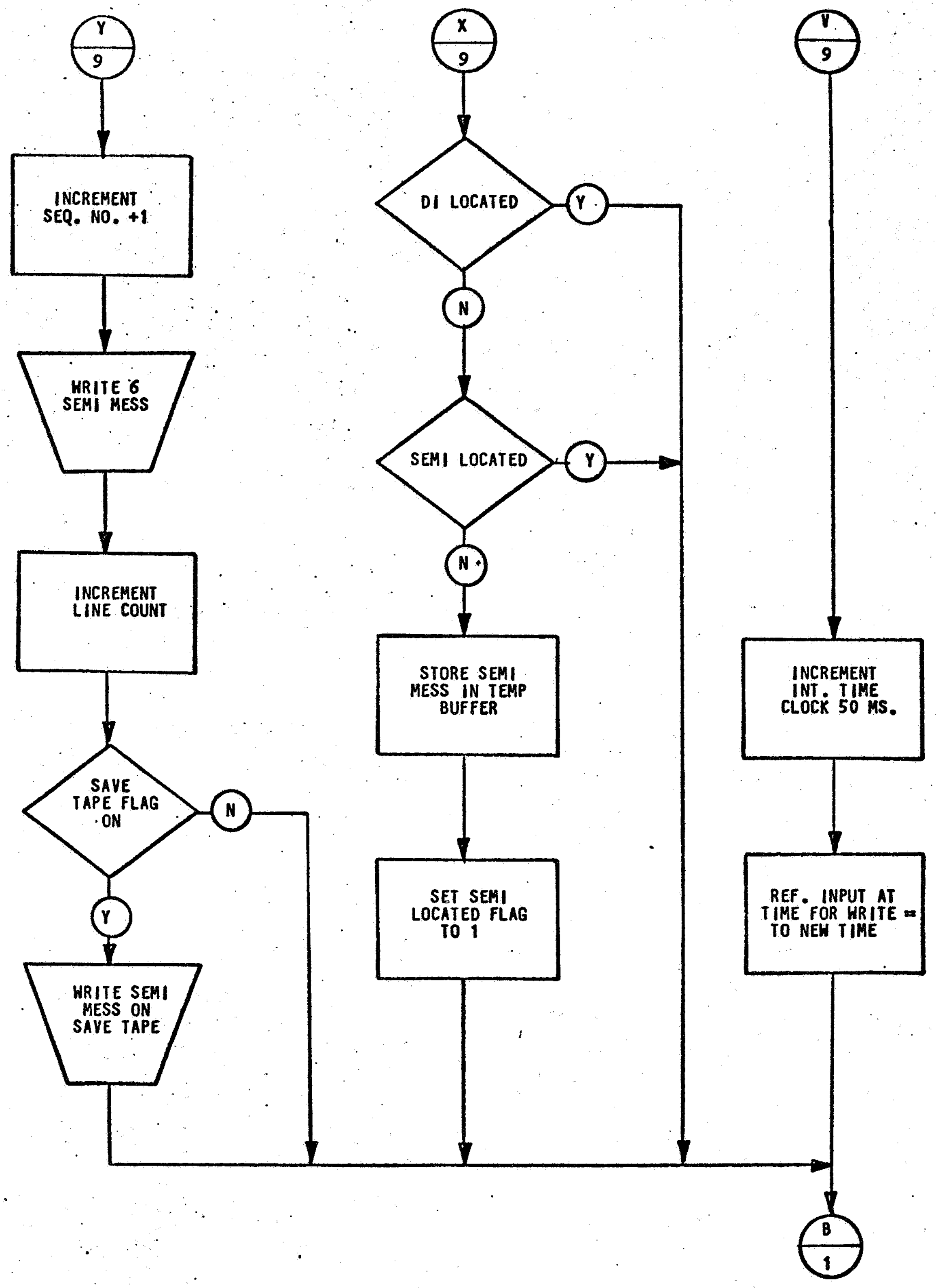

Y 9
X 9
V 9
INCREMENT SEQ. NO. +1
WRITE 6 SEMI MESS
INCREMENT LINE COUNT
SAVE TAPE FLAG ON
N
Y
WRITE SEMI MESS ON SAVE TAPE
DI LOCATED
Y
N
SEMI LOCATED
Y
N
STORE SEMI MESS IN TEMP BUFFER
SET SEMI LOCATED FLAG TO 1
INCREMENT INT. TIME CLOCK 50 MS.
REF. INPUT AT TIME FOR WRITE = TO NEW TIME
B 1

Figure A-2. SUBROUTINE KATLT (23 of 23)

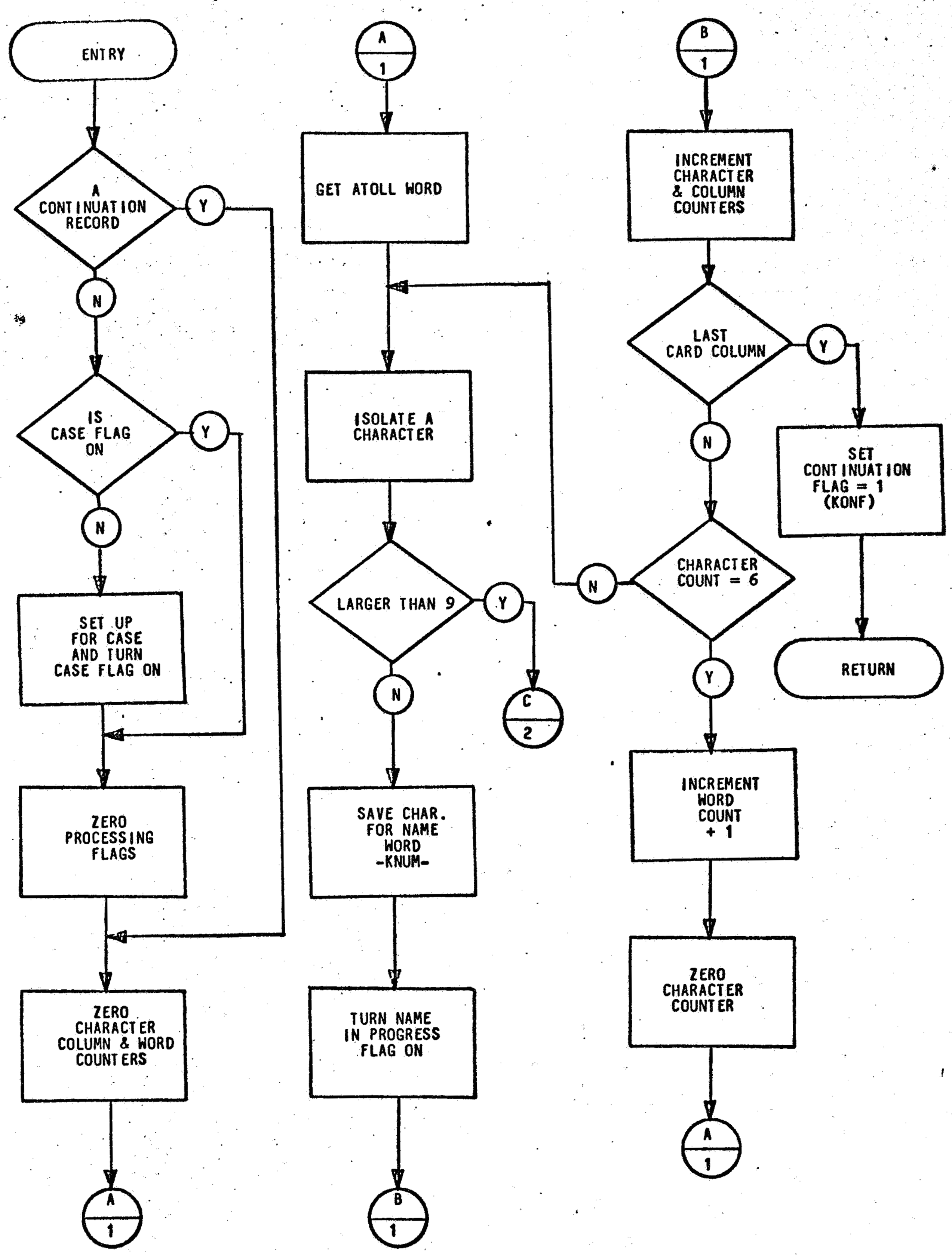


Figure A-3. SUBROUTINE KATRG (1 of 3)

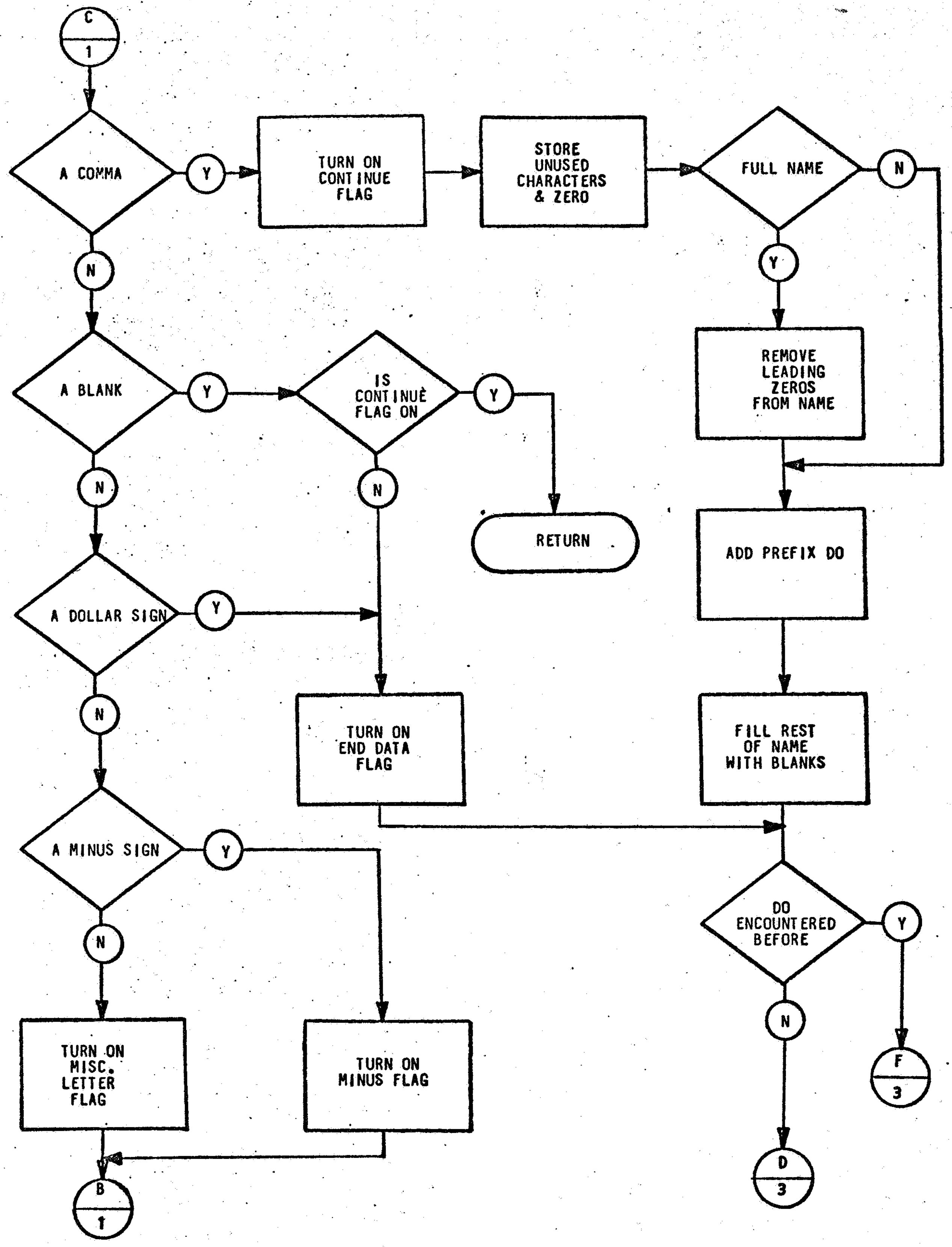


Figure A-3. SUBROUTINE KATRG (2 of 3)

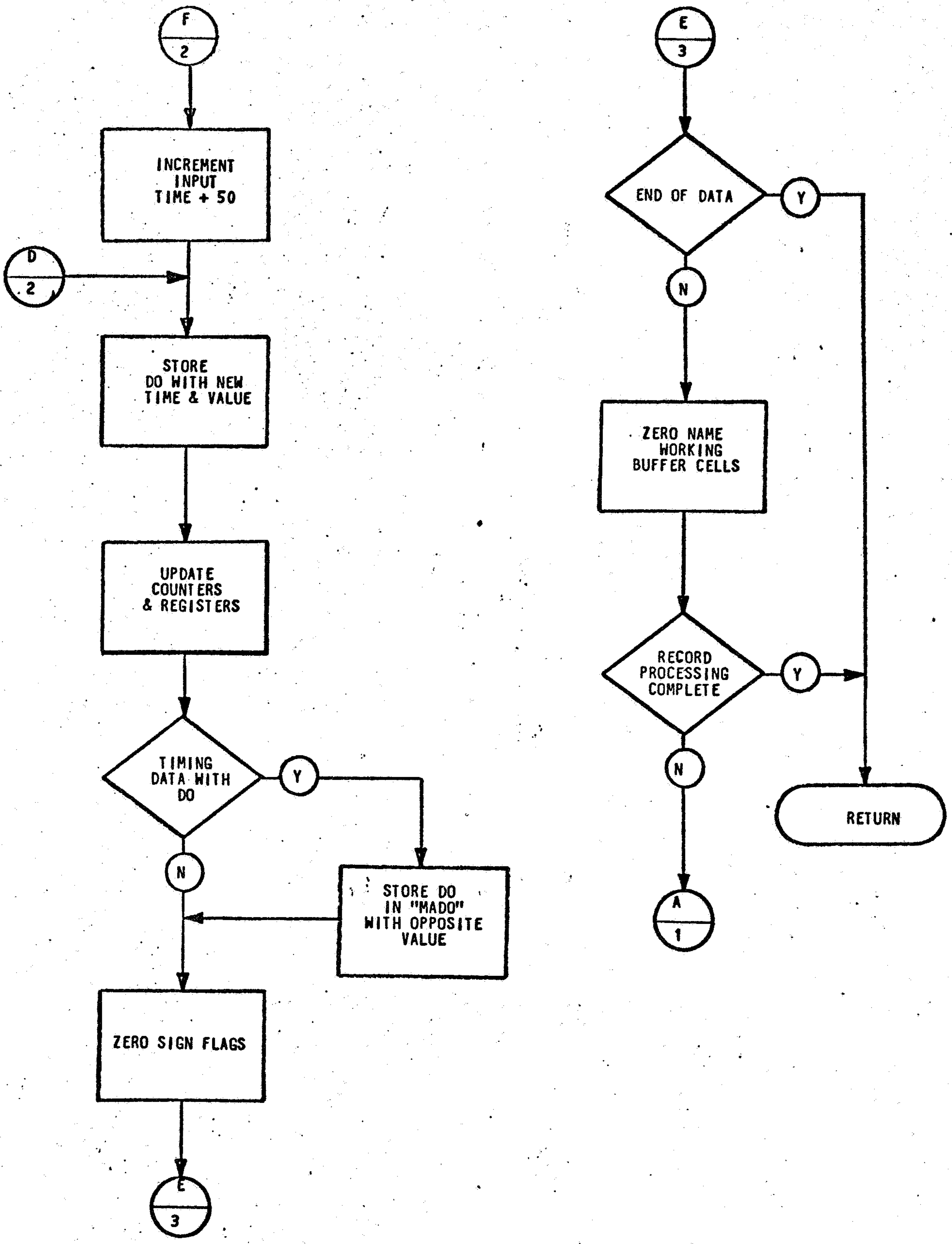

F
2
INCREMENT INPUT TIME + 50
D
2
STORE DO WITH NEW TIME & VALUE
UPDATE COUNTERS & REGISTERS
TIMING DATA WITH DO
Y
N
STORE DO IN "MADO" WITH OPPOSITE VALUE
ZERO SIGN FLAGS
E
3
E
3
END OF DATA
Y
N
ZERO NAME WORKING BUFFER CELLS
RECORD PROCESSING COMPLETE
Y
N
RETURN
A
1

Figure A-3. SUBROUTINE KATRG (3 of 3)

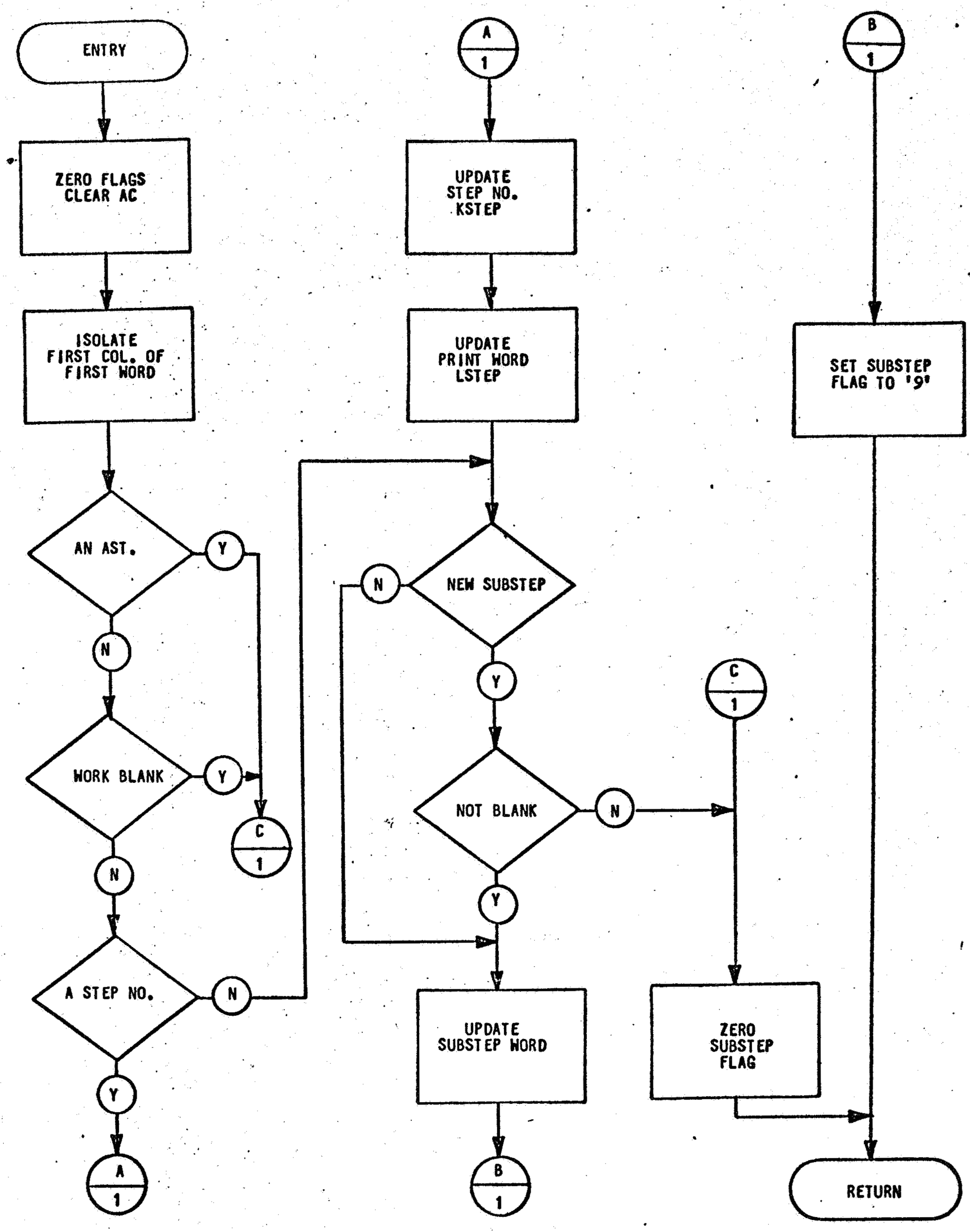


Figure A-4. SUBROUTINE KONVRT (1 of 1)

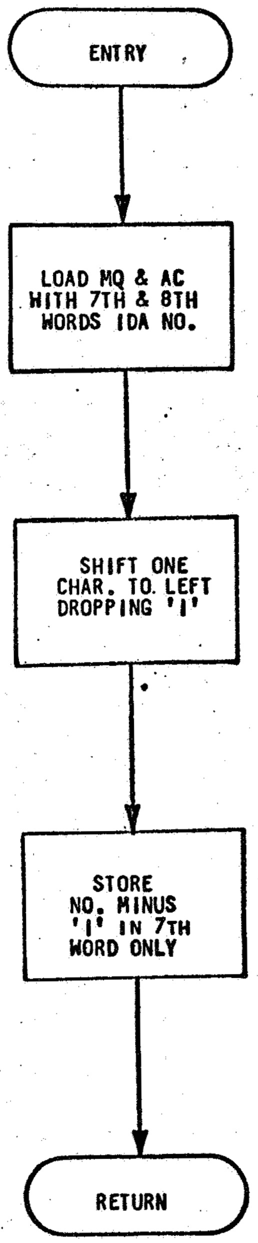

ENTRY
LOAD MQ & AC WITH 7TH & 8TH WORDS IDA NO.
SHIFT ONE CHAR. TO LEFT DROPPING 'I'
STORE NO. MINUS 'I' IN 7TH WORD ONLY
RETURN

Figure A-5. SUBROUTINE NIDEX (1 of 1)

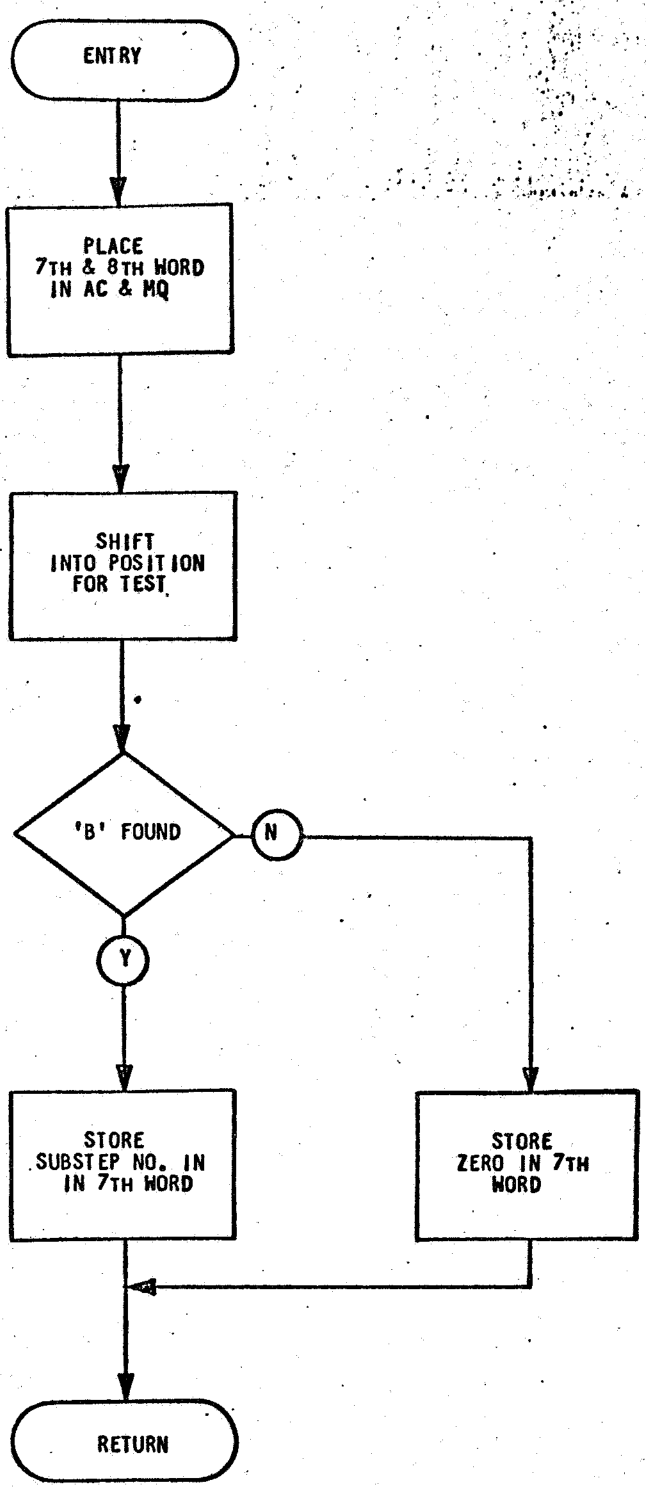

ENTRY
PLACE 7TH & 8TH WORD IN AC & MQ
SHIFT INTO POSITION FOR TEST
'B' FOUND
N
Y
STORE SUBSTEP NO. IN IN 7TH WORD
STORE ZERO IN 7TH WORD
RETURN

Figure A-6. SUBROUTINE NBRANZ (1 of 1)

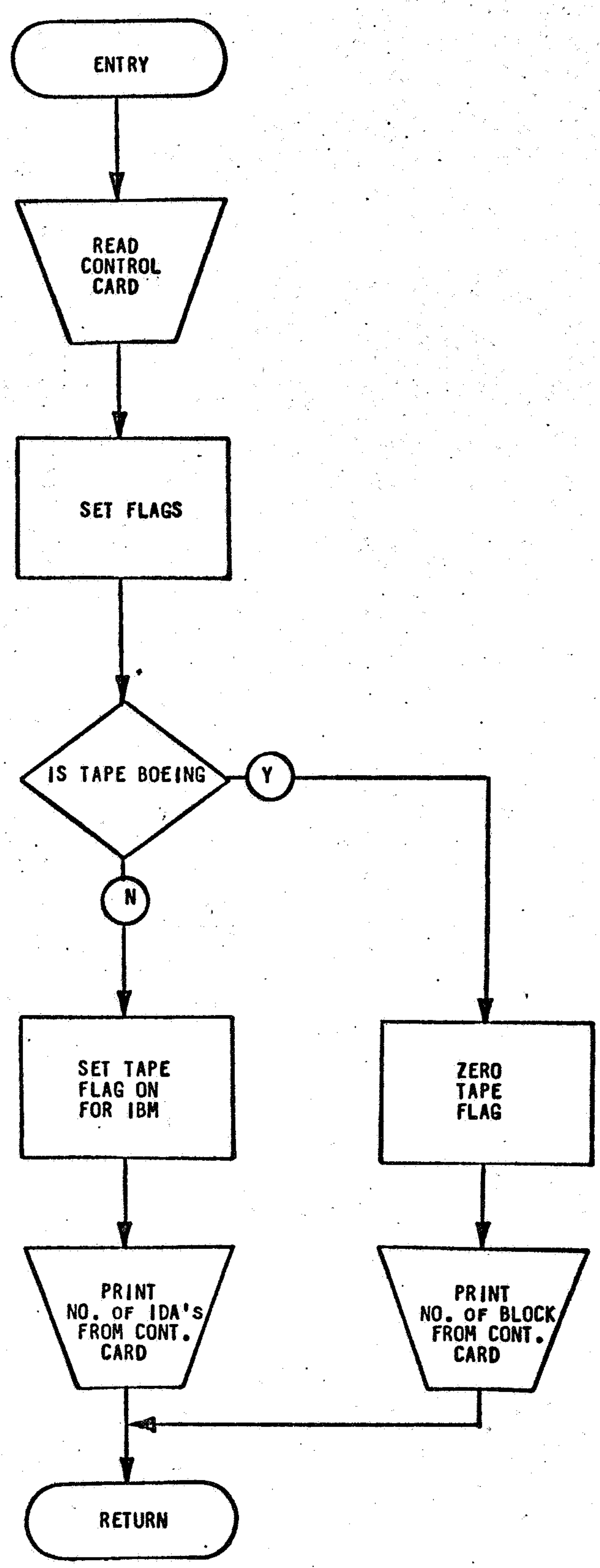


Figure A-7. SUBROUTINE READCO (1 of 1)

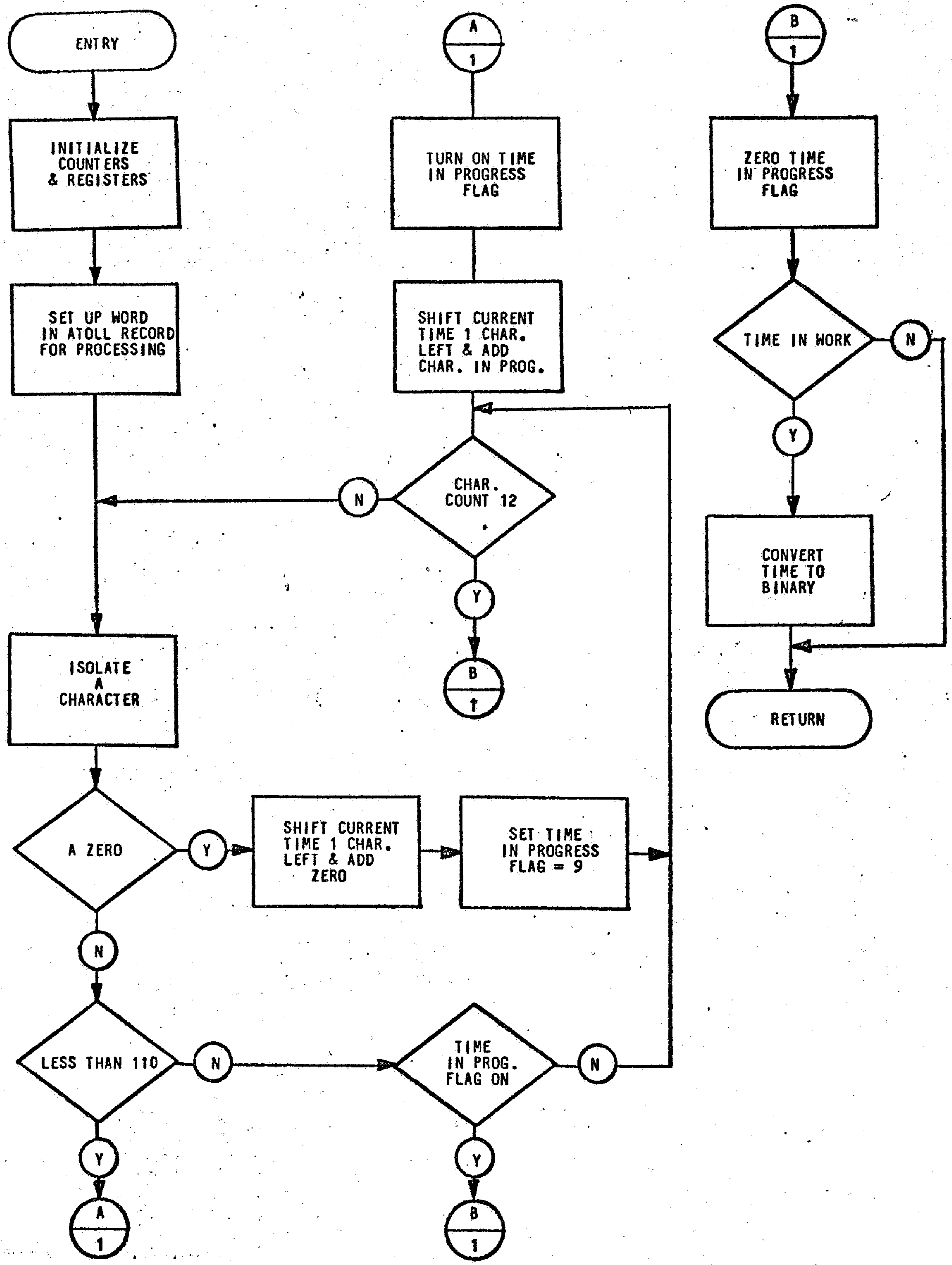

ENTRY
INITIALIZE COUNTERS & REGISTERS
SET UP WORD IN ATOLL RECORD FOR PROCESSING
ISOLATE A CHARACTER
A ZERO
Y
SHIFT CURRENT TIME 1 CHAR. LEFT & ADD ZERO
SET TIME IN PROGRESS FLAG = 9
N
LESS THAN 110
N
TIME IN PROG. FLAG ON
N
Y
Y
A 1
B 1
A 1
TURN ON TIME IN PROGRESS FLAG
SHIFT CURRENT TIME 1 CHAR. LEFT & ADD CHAR. IN PROG.
CHAR. COUNT 12
N
Y
B 1
B 1
ZERO TIME IN PROGRESS FLAG
TIME IN WORK
N
Y
CONVERT TIME TO BINARY
RETURN

Figure A-8. SUBROUTINE TIMEX (1 of 1)

PRECEDING PAGE BLANK NOT FILMED.

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLES

The following is an alphabetical listing of the terms used in the Input Conversion and Punch program.

NAME	DESCRIPTION
IBMF	Format flag.
IDA	Test number.
INTIM	Input time buffer.
JOPT	Scan timing code.
KADO	DO list.
KADT	Input time.
KADV	Input value.
KAFG	ATOLL file flag.
KAWDS	ATOLL input buffer.
KBUF	SEMI I/O buffer.
KCAS	Case count.
KDIF	Spare.
KDOF	DO value flag.
KDOS	Case input count.
KDIF	DO time flag.
KENDF	ATOLL end data flag.
KENF	ATOLL DI found flag.
KINF	ATOLL DO found flag.
KNUF	ATOLL name process flag.
KNUM	ATOLL DO name.
KONF	ATOLL record continue flag.
KRDF	ATOLL case flag.
KSEMI	SEMI found flag.
KSEQ	Sequence count.
KTIM	Time transfer word.
KTRL	Processing control flag.
KWD	Time format selector.
KYPE	Tape format flag.

NAME	DESCRIPTION
LCT	Print line count.
LIST	Save tape flag.
LSFG	Scan time flag.
LSTEP	Step number.
LSTIM	Case start time.
MADO	Cycled input list.
MADV	Number cycles.
MDOS	Cycled input count.
MNT	Number IDAS in run.
MSCAN	ATOLL scan duration.
MTIM	ATOLL DO duration.
MXTIM	Time correction factor.
NAM	Test name.
NBLOC	Block number.
NBRAN	NBRANZ call flag.
NCAS	IDA end flag.
NEXT	Test word.
NID	IDA number.
NPAG	Page count.
NSFG	Case flag.
NSTEP	Step number.
NXTB	Search count.

2.0 DEFINITIONS

NAME	DESCRIPTION
IBMF	A word in common block/ISPEC/ used by the program as a means of selecting alternate groups of instructions for processing certain ATOLL format or style differences between Boeing and IBM test procedures. The flag is set (by means of the control card) to 1 if the ATOLL tape to be used is an IBM tape. If not, the flag is zeroed.
IDA	A 10 word array in common block/ISPEC/ used to store any (up to 10) ATOLL test procedure names which are to be processed. The names (ID's) are stored here as 6 character BCD names exactly as they appear in the data field of the control card.

INTIM	A word in common block/MCAS/ used to store the current input time (simulation clock time) which will be assigned when a variable's input equation card is processed.
ISPEC	A common block of 14 words used to store the contents of the first control card. The array contains JOPT, KYPE, LIST, MNT, and 10 IDA locations.
JOPT	A word in common block/ISPEC/used for setting a value into LSFG if the timing data on ATOLL "SCAN" records is to be processed. The value is obtained in subroutine READCD when the control card is read.
KADO	A 20 word array in common block/KADOS/ used for storing the names of any ATOLL discrete outputs encountered when processing ATOLL "DISO 0" or "DISO 1" records during a case. The prefix "DO" is stored as two BCD characters left adjusted in the word. The numerical DO designation (from 1 to 4 BCD characters) is left adjusted to the prefix and the remainder of the word is filled with blanks. The array is zeroed at the start of each case.
KADOS	A common block containing counted KDOS, 20 KADO locations, 20 KADV locations, and 20 KADT locations used for storing ATOLL discrete output data during a case.
KADT	A 20 word array in common block/KADOS/ used for storing the simulation clock (input time) to be associated with a particular discrete name in array KADO. Each word is in binary format, and is zeroed at the start of each case.
KADV	A 20 word array in common block/KADOS/ used for storing the BCD value to be associated with a particular discrete name in array KADO. Each word used is in the format " =1 AT" or " = 0 AT", and is zeroed at the start of each case.

NAME	DESCRIPTION
KAFG	A word in common block/KNTRL/ used as a flag to signal that a particular test procedure is in process on the ATOLL tape. It is set when the particular test procedure name (NID) has been located on the ATOLL tape, and zeroed out when a subsequent ATOLL "END" record is encountered.
KATRL	A common block containing 8 words or flags used for controlling the reading and processing of the ATOLL tape. It contains KRDF, KONF, KENDF, KDOF, KDIF, KDTF, KNUM, KNUF.
KAWDS	A common block containing 14 words used as the input buffer for storing each 14 word ATOLL tape record as it is being processed.
KBOT	A test word in subroutine KATLT used for identifying ATOLL "BOTB" records.
KBUF	A 5 word array in subroutine KATLT used for temporary storage of the variables field of ATOLL "SEMI" records. It is used in conjunction with internal flag KSEMI, and is zeroed after use.
KCAS	A word in common block/MCAS/ used as a counter for storing the current number of ATOLL cases which have been processed. It is zeroed at the start of each test procedure.
KDIF	A spare location in common block/KATRL/ reserved for program expansion.
KDOF	A word in common block/KATRL/ used as a flag to signal the value of the current ATOLL discrete output associated with encountered ATOLL DISO 0 records, and is set plus for DISO 1 records.
KDOS	A word in common block/KADIS/ used as a counter for storing the number of ATOLL discrete outputs encountered during a case. It is zeroed at the start of each case.
KDTF	A word in common block/KATRL/ which is set in subroutine KATLT if timing data is encountered on

NAME	DESCRIPTION
	an ATOLL "DISO 0" or DISO 1" record. It is used to signal that the DO associated with that particular ATOLL operator is a "pulsed" input, and will therefore require an additional "restoring" input equation. The word is zeroed after the record has been converted and processed.
KEND	A test word in subroutine KATLT used for identifying ATOLL "END" records.
KENDF	A word in common block/KATRL/ used as a flag to signal that all names associated with a particular ATOLL "DISO 0" or "DISO 1" record have been processed. It is set whenever an ATOLL "end of variable field" is encountered or signaled.
KENF	A flag in subroutine KATLT used to indicate that an ATOLL "DISI 0" or "DISI 1" record has been encountered. It is zeroed at the end of each case.
KINF	An internal flag in subroutine KATOL. It is set to 1 if an ATOLL "DISO 0" or "DISO 1" record is encountered. It is tested when a case end is detected to determine how the current case is to be processed. It is zeroed at the start of a case.
KNAM	A test word in subroutine KATOL used to identify ATOLL "NAME" records.
KNT	An internal counter and test word in driver program CONTRD used to keep track of the number of test procedures remaining to be processed. It is set with the number from control card 1 (MNT); and tested before each test procedure is started. (If zero, the program exits. If not, KHT is decremented and a test procedure is set up for conversion).
KNTRL	A common block containing 5 process control flags (KTRL, IBMF, NSFG, KAFG, LSFG).
KNUF	A word in common block/KATRL/ used as a flag and temporary character storage cell during the identification of names encountered in the ATOLL variables

NAME	DESCRIPTION
	field. It is set in KATRG when any of the characters (0 thru 9) are detected, and zeroed whenever any other character is encountered.
KNUM	A word in common block/KATRL/ used as temporary storage while building up the numerical name designation for discrete predictions encountered while processing the variables field of ATOLL "DISO 0" or "DISO 1" records. The word is formatted in BCD, and will contain a 4 character numerical discrete designation right adjusted and zero filled. The word is reformatted before storing in array KADO.
KOLD	A location in subroutine KATRG used as temporary storage for the current word in the ATOLL variables field being processed.
KONF	A word in common block/KATRL/ used as a flag to signal that the current ATOLL variables field is to be continued on the next ATOLL record. It is set in subroutine KATRG if a continuation character is encountered, or no field termination characters are encountered. It is zeroed before each new variable field is processed.
KRDF	A word in common block/KATRL/ which is set in subroutine KATRG during the first entry in a case. It signals any subsequent entries that initialization of controls for discrete output data has been completed. It is zeroed at the end of each case.
KSEMI	A word in subroutine KATLT used as a flag to signal that the variables field of an ATOLL "SEMI" record has been stored in array KBUF for subsequent printout at the conclusion of a case. It is zeroed at the start of each case.
KSEQ	A word in subroutine KATLT used as a counter for storing the sequence number assigned to each input card processed by the program.

NAME	DESCRIPTION
KTIM	A word in common block/MCAS/ used as temporary storage by subroutine TIMEZ during the processing of the timing data encountered on ATOLL records. It is used to store the timing information converted to binary format, and is zeroed after return to the calling routine.
KTRL	A word in common block/KNTRL/ used as a processing status flag. Its value is returned to CONTRD as 2 for all normal returns. If its value is 10, a processing error has been detected, and the program is set to dump and exit.
KWD	A word in subroutine KATLT which is set to 1 if the accumulated internal "simulation clock" time INTIM (in milliseconds) exceeds 15 minutes. It is used as a key word for selecting an alternate format for processing any subsequent input equation cards.
KYPE	A word in common block/ISPEC/ used to signal the value to be assigned to ATOLL format flag IBMF. The value assigned to KYPE is taken from control card 1.
LCT	A word in subroutine KATLT used as a counter for the lines of printout on a page. If the line count exceeds 50 when printing the input equation, a new page is numbered and titled.
LETF	A word in subroutine KATRG used to store and signal when any character other than a number or a control character has been encountered while processing the variables field of an ATOLL record. It is zeroed whenever a number or a control character is encountered.
LIST	A word in common block/ISPEC/ used to signal the program that the converted inputs from the ATOLL tape are to be stored on an output save tape instead of the system punch tape. It is set in subroutine READCD when the control card is read.

NAME	DESCRIPTION
LSFG	A word in common block/KNTRL/ used in subroutine KATLT to signal that timing data encountered on ATOLL "SCAN" records will be processed and used to update the internal "simulation clock" INTIM. Its value is set in subroutine READCD from the contents of JOPT.
LSTEP	A word in common block/NSTEPS/ used in subroutine KONVRT to store the current test step number. It is formatted in BCD for use when printing out explanatory messages, and is taken from word 1 of each ATOLL record.
LSTIM	A word in subroutine KATLT used to test, update if required, and store the "simulation clock" INTIM at the start of a new case. The contents represent time in milliseconds and are in binary.
MADO	A 20 word array in common block/MADOS/ used as temporary storage for names of DO's which were indicated on the ATOLL record as being "pulsed" signals. Each word is formatted in BCD, and is transferred to array KADO before printout. MADO is zeroed at the start of each case, and is used in conjunction with counter MDOS.
MADV	A 20 word array in common block/MADOS/ used as temporary storage for the "restoring" value associated with a "pulsed" DO name placed in array MADO. The value is in BCD format, and is transferred to array KADV before printout. MADV is zeroed at the start of each case.
MCAS	A common block of 4 words containing NCAS, KCAS, INTIM, and KTIM. These words are used as case and time control words during the processing of a test procedure. They are zeroed at the start of each test procedure.
MDOS	A word in common block/MADOS/ used as a counter for storing the number of "pulsed" DO's encountered during a case.

NAME	DESCRIPTION
MNT	A word in common block/ISPEC/ used for storing the number of test procedures which are to be processed during the run. Its value is set in subroutine READCD when the control card is read.
MSCAN	A word in subroutine KATLT used as a flag and temporary storage for timing data encountered with ATOLL "SCAN" operators if the flag LSFG has been set. If LSFC is set, any timing data which is identified (by subroutine TIMEZ) is placed in MSCAN for updating the current "simulation clock" INTIM. MSCAN is zeroed after use.
MSIGN	A word in subroutine KATRG used as an internal flag to signal that the current discrete name being processed was preceeded by a minus sign. (The value assigned to this particular DO will be reversed from the value signaled by the ATOLL operator).
MTIM	A word in subroutine KATLT used as a flag and temporary storage for pulse timing data encountered with ATOLL "DISO 0" or "DISO 1" records. Any timing data encountered for DO's which are "pulsed" signals is converted in subroutine TIMEZ and placed in MTIM for updating the "simulation clock" when the DO "restoring" input equations are processed. MTIM is zeroed after use.
MXTIM	A word in subroutine KATLT which contains the fixed octal equivalent of 900,000 milliseconds (15 minutes). It is used as a test word for determining if the accumulated "simulation clock" INTIM reaches this value during the test procedure. If so, an alternate simulation input format is used, and INTIM is restarted from zero.
NAM	A word in common block/NAMS/ used for storing the name of the current test procedure. It is set in subroutine DNSINP, and is formatted in BCD for use when printing out the title heading.

NAME	DESCRIPTION
NAMS	A common block containing 4 storage locations for test procedure identification and status data. It contains NID, NAM, NBLOC, and NXTB.
NBLOC	A word in common block/NAMS/ used for storing the current test block number. Its value is set in subroutine KATLT with data from the variables field of ATOLL "BOTB" records, and is formatted in BCD for use when printing out the equivalent simulation "BLOCK" inputs.
NBRAN	The Call for subroutine NBRANZ used in subroutine KATLT when processing ATOLL "test" records with branching data.
NCAS	A word in common block/MCAS/ used as a flag for storing the current status of program processing. Its value is set to 2 in subroutine KATLT to signal that the current ATOLL procedure "END" record has been encountered and all inputs have been processed. It is zeroed at the start of each test procedure.
NEXT	A word in common block/NSTEPS/ used as a flag to signal that the first word of the current ATOLL record contains a step or substep number. It is set to 9 in subroutine KONVRT if step-substep data is present. If the first word is blank, or is a comment, NEXT is set to zero.
NID	A word in common block/NAMS/ used to store the identification word for the current ATOLL test procedure in process. Data for the word is obtained from the current position in array IDA, and is formatted in BCD.
NPAG	A word in subroutine KATLT used as a counter for numbering pages when processing program results for standard output printing. It is zeroed at the start of each test procedure conversion.
NSFG	A word in common block/KNTRL/ used as a flag in subroutine KATLT to signal that a simulation "STEP" input card has been processed and initialization for

NAME	DESCRIPTION
	the start of a new case has been completed. It is zeroed at the start of a case, and is set when the first ATOLL, DISO, SEMI, or TEST INSTRUCTION is encountered.
NSTEP	A word in common block/NSTEPS/ used for storing the combined step/substep corresponding to the current ATOLL record being processed. The word is in BCD format and is setup in subroutine KONVRT. The word is used in processing the SIMULATION "STEP" records. (These become the correlating links when the processed inputs are used to drive a DNS model of the system under test and later for comparing the results of the simulation with the predictions on the ATOLL tape).
NSTEPS	A 3 word common block containing NSTEP, NEXT, and LSTEP. It is used for storing test procedure step and substep data.
NXTB	A word in common block/NAMS/ used as a counter in subroutine KATLT when searching the ATOLL tape for a current test procedure name. If the count exceeds 10, processing is discontinued and KTRL is set to 10 to signal a DUMP and EXIT.

SECTION 4

DNS COMPARATOR PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

4

DNS/ATOLL COMPARATOR PROGRAM

AUTHOR: A. R. Stone
Convair division of General Dynamics
Huntsville Operations
11 November 1967

PURPOSE: THE DNS/ATOLL Comparator Program was developed as part of a test procedure validation technique based on Discrete Network Simulation (DNS). The Program correlates and compares the results contained on the output tape from a test procedure Simulation with the equivalent discrete predictions contained on the ATOLL card image tape for the particular test procedure. It produces a listing of discrepancies identified to particular test steps, and a listing of any modeled components in the system which were not actuated or exercised during the test.

RESTRICTIONS:

1. The program must run on an IBM 7094 with IBJOB systems capability.

2. In addition to system input and output, two magnetic tape units are required for BCD input tapes.

3. A maximum of ten tests may be processed during one computer run.

4. The maximum number of discrete predictions in any one step is limited to 100.

STORAGE: The program and its associated buffer storage area extends consecutively from core location 3046)8 to 25777)8. The program consists of the following fourteen subprograms:

1.	CONTRD	Driver (Fortran IV).
2.	CONVRT	Formats step and substep data (MAP).
3.	DNSINP	Reads and identifies records on simulation output tape (Fortran IV).
4.	DNSRDG	Identifies, processes, and stores discrete inputs and values from the simulation (MAP).
5.	DOSRDG	Identifies and stores any redundant discrete outputs from the simulation. (MAP).
6.	KATLT	Reads and identifies records on the ATOLL card image tape. (Fortran IV).
7.	KATRG	Identifies, processes, and stores names and values of ATOLL discrete predictions (MAP).
8.	KEPCKZ	Identifies and updates step/substep key word for DNS and ATOLL tape correlation (MAP).
9.	CHEKZ	Controls data comparison and preparation of results for printing (Fortran IV).
10.	COMPAZ	Compares names and values of discretes and identifies any differences (Fortran IV).
11.	NIDEX	Reformats test procedure identification for DNS/ATOLL compatibility (MAP).
12.	PREPAZ	Reconstructs and formats discrete names and values for printout (Fortran IV).
13.	PRINZ	Prints discrepancy headings, discrepancy lists, and messages. (Fortran IV).
14.	READCD	Reads program data control cards (Fortran IV).

TIMING: The program processes approximately 3000 tape records (1000 ATOLL and 2000 DNS tape records) containing an average of 120 ATOLL steps and 60 DNS case histories per minute.

USE: A run request, two magnetic tapes, and two program control cards are required to set up the program for operation. The two particular tapes to be compared will be designated on the run request. The first of these tapes is an ATOLL BCD card image tape for a selected test procedure, or group of test procedures. The second tape is a simulation BCD output tape previously created by stimulating a DNS model of the system under test with driving functions derived from the ATOLL card image tape. (Refer to Section 3, DNS/ATOLL Input Conversion and Punch Program). A typical operational card deck setup is:

```
$JOB
$PAUSE          Mount tapes.
$ATTACH         A5   (ATOLL card image)
$AS             SYSCK1,HI
$ATTACH         B6   (simulation output)
$AS             SYSUT6,HI
$EXECUTE        IBJOB
$IBJOB          GO
(DNS/ATOLL Comparator Program Binary Deck)
$DATA
CONTROL CARD 1
CONTROL CARD 2
(Card 2 continuation cards if required)
7/8 end of file
```

Control card usage is as follows:

Card 1 - Job specification card always required.

Col. 3	Blank (unused spare)
Col. 4, 5	Blank
Col. 6	1 or 2 (to designate type of ATOLL tape - 1 is for IBM ATOLL format, 2 is for Boeing format).
Col. 7,8	Blank
Col. 9	0 or 1 (to specify whether simulation state lists are to be processed - if 0, skip; if 1, process).
Col. 10 thru 12	Integer number between 1 and 10 to designate the number of test procedures to be compared during this run.
Col. 13 thru 18	Identification number of first test procedure to be processed. The identification will be formatted the same as identification contained on the simulation output tape.
Col. 19 thru 25	Identification of second test procedure if included.
Col. 26 thru 72	Up to 10 additional test ID's if needed.

Card 2 - Data definition - always required.

Col. 1 thru 4	A number between 0 and 200, right adjusted, to designate the number, if any, of ATOLL discrete inputs which are to be excluded from the comparison. If none are to be excluded, Col. 4 must contain a 0.
(Col. 5 thru 80)	(Blank if Col. 4 is 0)
Col. 5 thru 8	(Numeric identifier of first DI,
9 12	2nd DI, etc., depending on number of
13 16	discretes designated, and extending from Cols. 0 thru 80 of succeeding cards as required to list the required number of DI numerical identifiers.)

METHOD:

A. The program first reads the two control cards. It utilizes the first control card to determine the type of ATOLL format it will encounter, the test procedures to be processed, and whether any unused components are to be listed at the end of the test comparison. The second control card signals whether any discrete inputs (DI's) are to be excluded from comparison (such as "Don't care DI's", or external discretes not incorporated into the DNS model). If DI's are to be excluded, the number and designations of the DI's are read, formatted, and stored in an array for matching against DI's encountered as the ATOLL predictions are read. Counters for the number of test procedures remaining to be processed are initialized.

B. The number of test procedures remaining to be processed is checked. If all have been processed, the program cleans up and exits. If test procedures remain to be processed, process counters are updated and a corresponding test procedure ID from the input list is placed in the current name cell.

1. Records on the simulation output tape are checked against the current name cell until the matching ID word is located. When located, process controls are set to signal that the simulation tape is in position to start the test procedure comparison.

(If not located, controls are set to signal unexpected end of simulation tape, and a dump and exit is made).

2. Records on the ATOLL tape are then checked against the current name cell until the matching ID word is again located. When located, process controls are set to signal that the ATOLL tape is also in position to start the test procedure comparison. (If not located, an unexpected end of file will be encountered, and the run will be discontinued).

C. Storage areas for encountered simulation data are initialized, and a simulation case history is read off the simulation tape.

1. The simulation case comprises all data from the first "step" record encountered until a new (next) "step" record (or an "end" record) is read. The actual step and substep numbers contained on these "step" records become the unique case starting and case ending correlation words for locating the equivalent data records on the ATOLL tape. The case ending "step" record is held in standby, and will become the starting "step" for the next case.

2. The records following the first "step" record are checked for specified simulation activity labels, and in turn for discrete data. Any discrete inputs encountered are then checked and processed as follows:

   a. If the DI has not been encountered previously during the current case, the DI value and the numerical designation are combined and stored as a unique word in a simulation DI array NSDI.

   b. If a DI has been encountered previously during the current case, it is stored in a recurring DI array NYCLE. If the DI value has changed, the new value is substituted for the previous value in the appropriate position in the simulation DI array NSDI.

c. If a DO is encountered on a record with the DNS "extra" label, the DO designation is stored in a "redundant input" array, NSDO.

d. When the next "step" record (or "end" record) is encountered, the current case is concluded, and controls are transferred for locating and processing the equivalent data on the ATOLL tape.

D. Storage areas for encountered ATOLL data are initialized and the ATOLL record containing the unique case started correlation word (refer to C. 1) is located. The case starting record and all subsequent records are checked for specified ATOLL operators, for step and substep numbers, and any discrete prediction data is processed until a record containing the case ending correlation word is encountered. The case ending record is held in standby and in turn becomes the starting record for the next case. The test and DI prediction data encountered are processed as follows:

1. If a pre-specified list of DI's to be ignored was included at run time, each DI encountered during the case is checked against this list. If found in the list, the DI is merely stored in a "Don't Care" array LOST for subsequent comments listing. If not found in the list, the DI is processed exactly as in C.2.a and C.2.b, except that the processed data is stored in ATOLL counterparts KADI and KYCLE of the Simulation arrays.

2. Test operators encountered are checked for branching data. Any steps and substeps specified as branch points are processed and printed as they occur, proceeded by a case heading identified to the starting step and substep.

3. When the case ending correlation word is encountered, the current case is concluded, and controls are transferred for comparing the ATOLL data with the Simulation data compiled from the current case.

E. The Simulation data and ATOLL data compiled from the current case are compared. If no differences are encountered, controls are transferred to F. If differences or discrepancies were encountered, and a case heading has not been processed (ref. D.2), a case heading identified to the starting step and substep is printed. Comparison of the Simulation data and ATOLL data proceeds as follows:

1. If any DO's were placed in the "redundant input" array, they are listed out under an explanatory heading.

2. Error flags and difference counters are initialized, and any combination "DI and value" words in the Simulation DI array NSDI are compared with any combination "DI and value" words in the ATOLL DI array KADI. All matching words are zeroed. Any remaining words in the Simulation DI array will have been different either in name, in value, or both, and are therefore formatted and printed out in conjunction with an explanatory heading. Similarly, any remaining words in the ATOLL DI array are printed out under an equivalent heading.

3. Error flags and different counters are re-initialized, and any recurring Simulation DI's are compared to any recurring ATOLL DI's. Any Simulation differences or ATOLL differences are printed out under explanatory headings.

4. If any DI's were placed in the "Don't Care" array LOST during the case, (ref. D.1), these DI's are printed out under an explanatory comment.

F. After completion of the current case comparison, the status of the processing is checked.

1. If the "end" (end of current test procedure) records have not occurred, control is transferred back to C. for reading in a new case.

2. If the "end" records have occurred, and the simulation cycles list is to be ignored, control is transferred back to B.

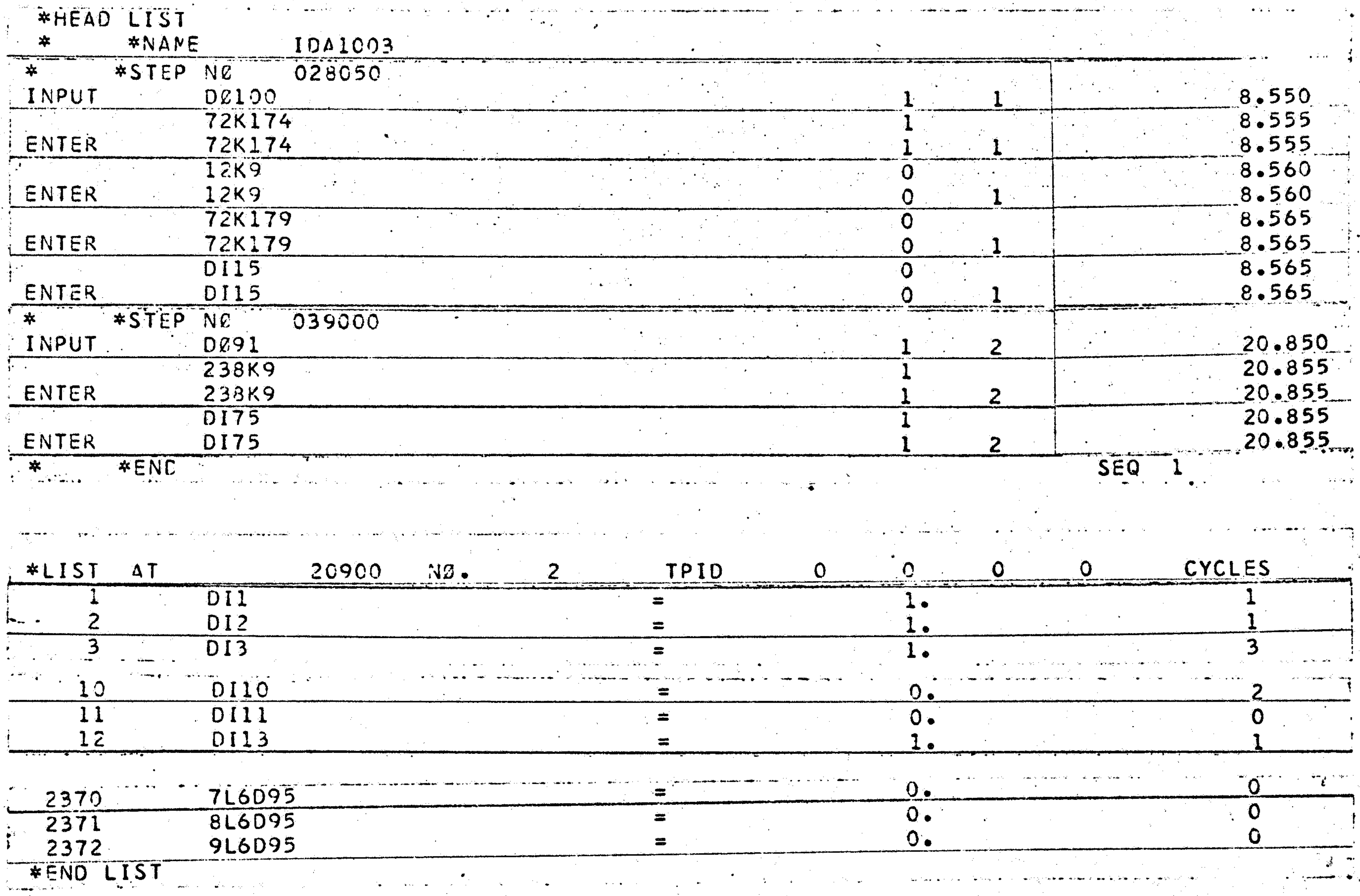
3. If the list is to be checked, the program proceeds to locate the DNS "*List" label. All records are checked for names and values contained in their cycle count. Any names encountered which were not activated or cycled during the test are listed out. When the end label of the list is encountered, control is returned to B.

INPUT FORMATS:

Figs. 4-1 and 4-2 illustrate the typical content and format of the two types of data to be compared by this program. The formats for the two control cards required in the input deck are discussed in the section headed "Use".

Fig. 4-1 shows representative portions of the results of a test procedure simulation. The portions shown are typical of the content and format of the simulation output tape. The lines identified on the left with an asterisk were the result of a simulation control card. The lines identified on the left by the activity label "input" were the result of an input equation. (These are reflections of the driving functions derived from the tape of Fig. 4-2 by the Input Conversion and Punch Program prior to the simulation). The remaining lines above the "LIST" label are typical of the history of reactions occurring in the model as a result of these inputs. Activity labels ('input' and 'entry') preceded and followed by an asterisk label constitute a 'case'. The case is identified by the step number on the asterisk label immediately preceding the activity label. The step number is used to correlate the simulation data with the equivalent ATOLL data. The lines below the "*LIST" label represent portions of the state list which records the final value for each variable, and how many times it changed state during the test procedure. The numbers at the left of these records are the internal code numbers of the variables, and are always listed in numerical order.

Fig. 4-2 contains representative portions of the ATOLL card image tape. The portions shown typify the IBM ATOLL format and data content. The program monitors all ATOLL instructions shown except the "Delay" and "Scan" instructions. Primary concern is with the 'DISI0' and 'DISI1' records, which contain the names and values of the predictions. The step and substep numbers which appear at the



```
*HEAD LIST
 *      *NAME       IDA1003
*      *STEP NØ     028050
INPUT        DØ100                                      1     1           8.550
             72K174                                     1                 8.555
ENTER        72K174                                     1     1           8.555
             12K9                                       0                 8.560
ENTER        12K9                                       0     1           8.560
             72K179                                     0                 8.565
ENTER        72K179                                     0     1           8.565
             DI15                                       0                 8.565
ENTER        DI15                                       0     1           8.565
*      *STEP NØ     039000
INPUT        DØ91                                       1     2          20.850
             238K9                                      1                20.855
ENTER        238K9                                      1     2          20.855
             DI75                                       1                20.855
ENTER        DI75                                       1     2          20.855
*      *END                                                       SEQ  1

*LIST  ΔT            20900   NØ.     2        TPID     0    0    0    0     CYCLES
     1       DI1                      =               1.                      1
     2       DI2                      =               1.                      1
     3       DI3                      =               1.                      3
    10       DI10                     =               0.                      2
    11       DI11                     =               0.                      0
    12       DI13                     =               1.                      1
  2370       7L6D95                   =               0.                      0
  2371       8L6D95                   =               0.                      0
  2372       9L6D95                   =               0.                      0
*END LIST
```

Figure 4-1. Representative parts of test procedure simulation.

```
 C0100NAME                                     IDA1003                    AU000200
  S-IU-209  E.D.S. SYSTEM TEST                                            AU000300
 28000DISØ1                                    DØN0251                    AU069600
      10DISØ1                                  DØN0030                    AU070600

      25DISI1                                  DIN0240                    AU071900
      30DISI0                                  DIN0230                    AU072000
      35DISI1                                  DIN0013                    AU072100
      40DELY                        1000                                  AU072200
      45SCAN                                                              AU072300
      50DISØ1                                  DØN0100                    AU072400
      55DISI0                                  DIN0015                    AU072500
      60SCAN                        250                                   AU072600
 28500TESTI                                    B031000,DIN1006            AU072700
 29000DISØ1                                    DØN0245                    AU072800
      10DISI0                                  DIN0226                    AU072900
      15DISI1                                  DIN0225                    AU073000

      25SCAN                                                              AU073800
 31000DISØ1                                    DØN0101                    AU073900


 39000DISØ1                                    DØN0091                    AU079900
      05DISI1                                  DIN0075$                   AU080000
      10SCAN                                                              AU080100
 44444END                                                                 AU080200
```

Figure 4-2. Representative parts of ATOLL card image tape.

start of each record are the basis for correlating this data with the simulation data. The step number occupies the first 4 characters in the record and is right adjusted. (The particular step numbers shown here are preceded by a blank. The next 2 characters constitute the substep number. The step number is not repeated, but is implied for succeeding substep numbers until a new step number is encountered).

OUTPUT FORMAT:

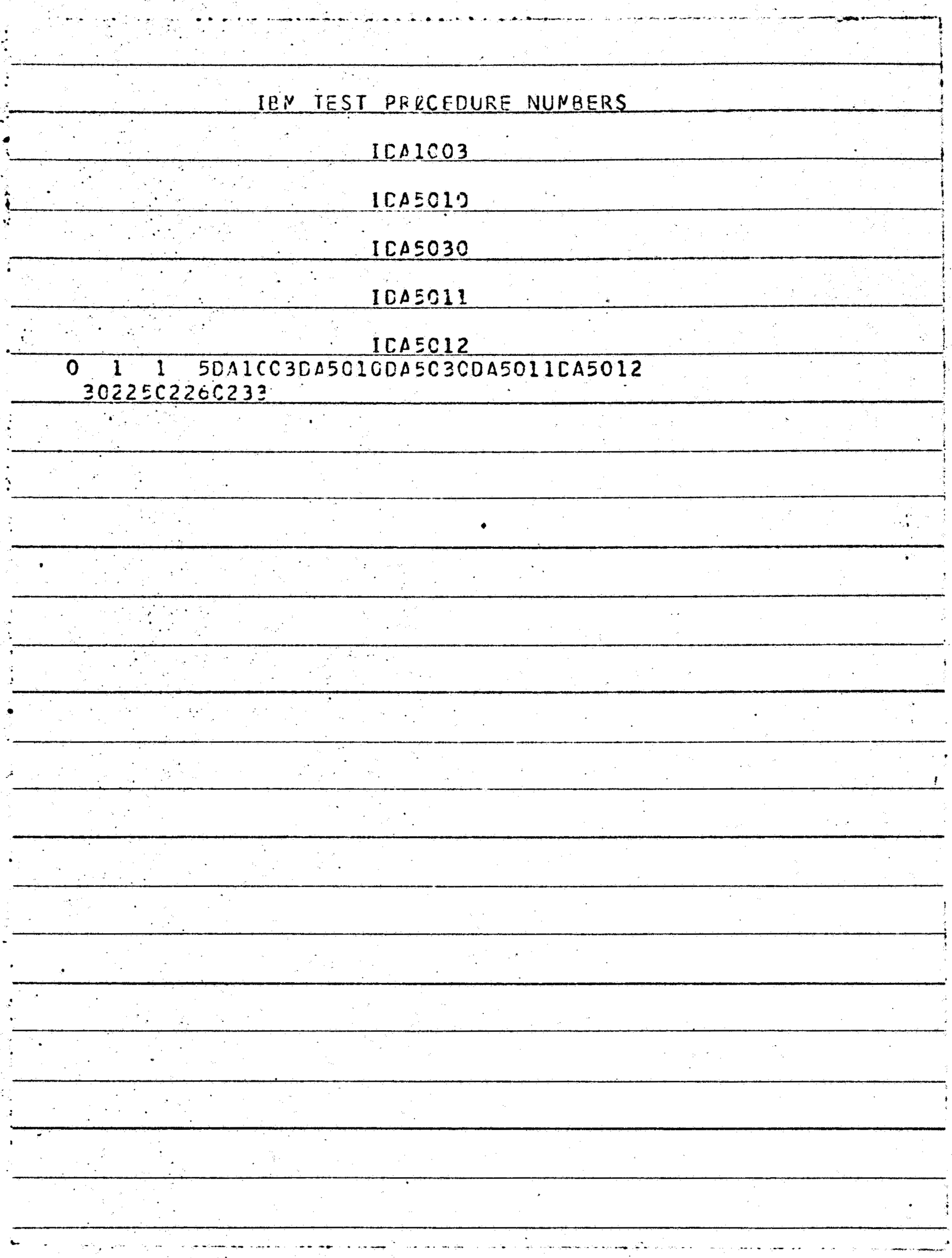
Fig. 4-3 represents the prologue printout from a comparison run. The bottom pair of lines are images of the two control cards required for the input deck. These cards were discussed in the section headed "USE". The preceding heading and test procedure list were printed by the program from the data in the first control card image. The second control card image signifies that 3 discretes, DI0225, DI0226, and DI0233 are to be excluded from any comparison.

Fig. 4-4 represents some typical discrepancy listings prepared by the program after conducting a comparison.

1. The message listed under step 170 indicates that simulation result DI267 was not found on any of the ATOLL "DISI1" records encountered during the case. (If DI267 had been located as a "DISI1" prediction, no step heading or message would have been processed. If DI267 had been found on a "DISI0" prediction, the heading "ATOLL Discrete Not in Simulation" and statement "DI0267 =0", would also have been included for this case).

2. The note listed for the case which started at step 0190 signifies that when DO121 had been processed as an input during the simulation run, its value was already at the state requested. This indicates that D0121 may be an unnecessary or redundant step in the test procedure.

3. The message listed under step 0280 substep 00 indicates that DI240, which was encountered on a "DISI 1" record (reference Fig. 4-2, DIN0240), was not found as a simulation result. The analysis here is handled as in 1. above.



```
            IBM TEST PRØCEDURE NUMBERS

                        ICA1003

                        ICA5010

                        ICA5030

                        ICA5011

                        ICA5012
  0   1   1   5DA1003DA5010DA5030DA5011CA5012
   3022502260233
```

Figure 4-3. Prologue printout for comparator program.

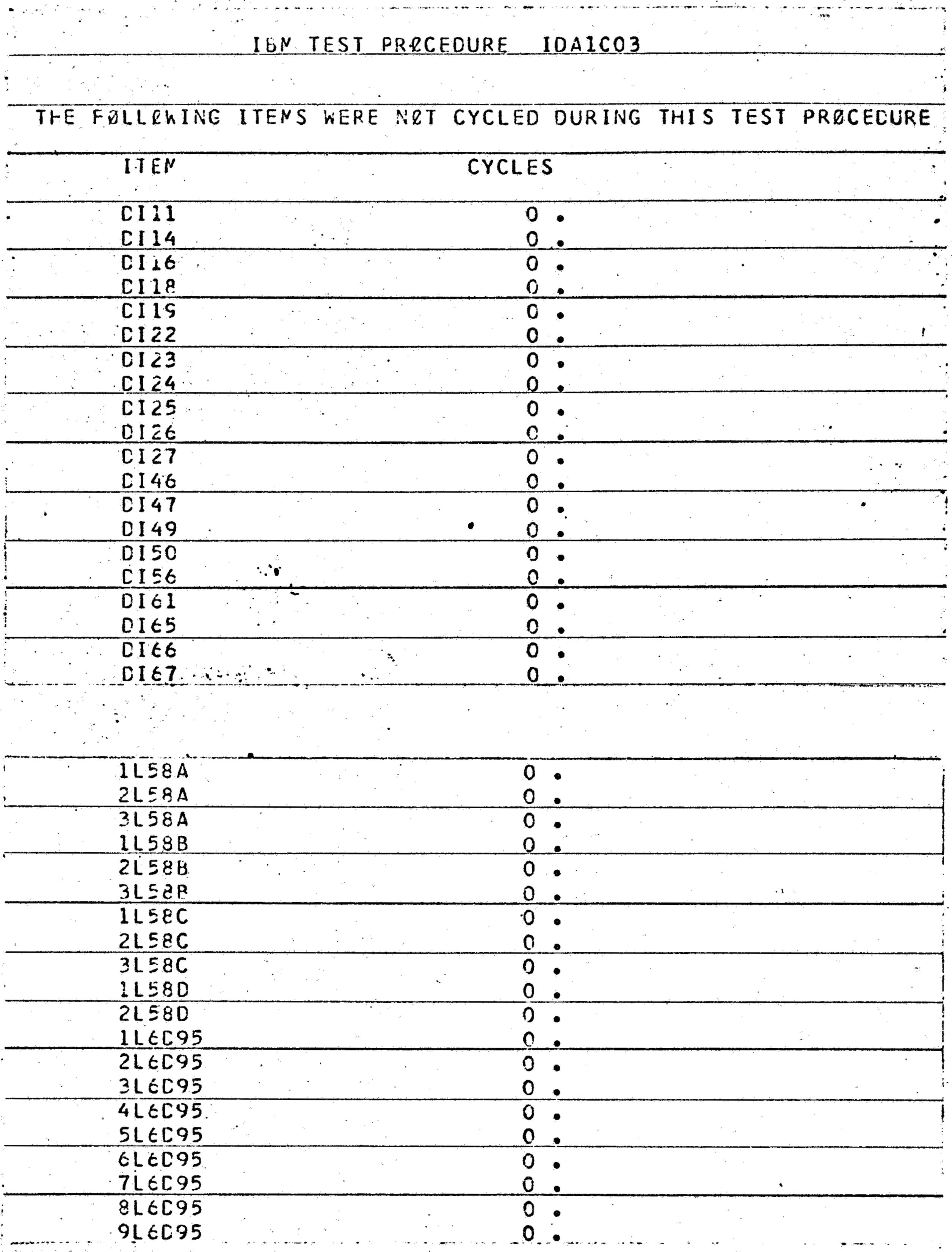
```
DISCRETE NETWØRK SIMULATIØN VS ATØLL TEST PRØCEDURE CØMPARISØN

    IBM TEST PRØCEDURE  IDA1C03

  STEP  0170        SUBSTEP  0C

      DNS DISCRETE IN NØT IN ATØLL
            DIC267         = 1.

  STEP  0190        SUBSTEP  0C


       NØTE - DISCRETE ØUT ALREADY AT THE STATE REQUESTED.
            DØC121



  STEP  0280        SUBSTEP  0C

      ATØLL DISCRETE IN NØT IN SIMULATIØN
            DIC240         = 1.


  STEP  0280        SUBSTEP  50

       NØTE - TEST AT STEP 0285 SUBSTEP 0C CAN BRANCH TØ 031000

       NØTE - ATØLL DISCRETE IS NØT IN DNS MØDEL
            DIC226
            DIC225


  STEP  0310        SUBSTEP  00

       NØTE   - DNS DISCRETE IN APPEARS MØRE THAN ØNCE.
            DIC121
            DIC122
            DIC123
```

Figure 4-4. Discrepancy listing for comparator program.

4. The first note listed under step 0280 substep 50 indicates that an ATOLL "Test" Record' shown on Fig. 4-2 was encountered. The second note signifies that DI0226 and DI0227 had been found in the data of Fig. 4-2, but had been included as ATOLL data to be ignored (reference Fig. 4-1).

5. The note listed under step 0310 signifies that three discretes cycled during the simulation before attaining final values which corroborated the ATOLL predictions. If corroboration had not occurred, additional messages would be included as in 1. above.

6. Fig. 4-5 represents a portion of the complete list of modelled variables or items which were not used in the test procedure. The list is obtained by lifting out of the Simulation state list (reference Fig. 4-1) all variables whose cycle count was zero at the end of the test.



```
             IBM TEST PRØCEDURE  IDA1C03

 THE FØLLØWING ITEMS WERE NØT CYCLED DURING THIS TEST PRØCEDURE

       ITEM                         CYCLES

       DI11                            0.
       DI14                            0.
       DI16                            0.
       DI18                            0.
       DI19                            0.
       DI22                            0.
       DI23                            0.
       DI24                            0.
       DI25                            0.
       DI26                            0.
       DI27                            0.
       DI46                            0.
       DI47                            0.
       DI49                            0.
       DI50                            0.
       DI56                            0.
       DI61                            0.
       DI65                            0.
       DI66                            0.
       DI67                            0.


       1L58A                           0.
       2L58A                           0.
       3L58A                           0.
       1L58B                           0.
       2L58B                           0.
       3L58B                           0.
       1L58C                           0.
       2L58C                           0.
       3L58C                           0.
       1L58D                           0.
       2L58D                           0.
       1L6D95                          0.
       2L6D95                          0.
       3L6D95                          0.
       4L6D95                          0.
       5L6D95                          0.
       6L6D95                          0.
       7L6D95                          0.
       8L6D95                          0.
       9L6D95                          0.
```

Figure 4-5. Unused component list from comparator program.

APPENDIX A

PROGRAM FLOW CHARTS

PRECEDING PAGE BLANK NOT FILMED.

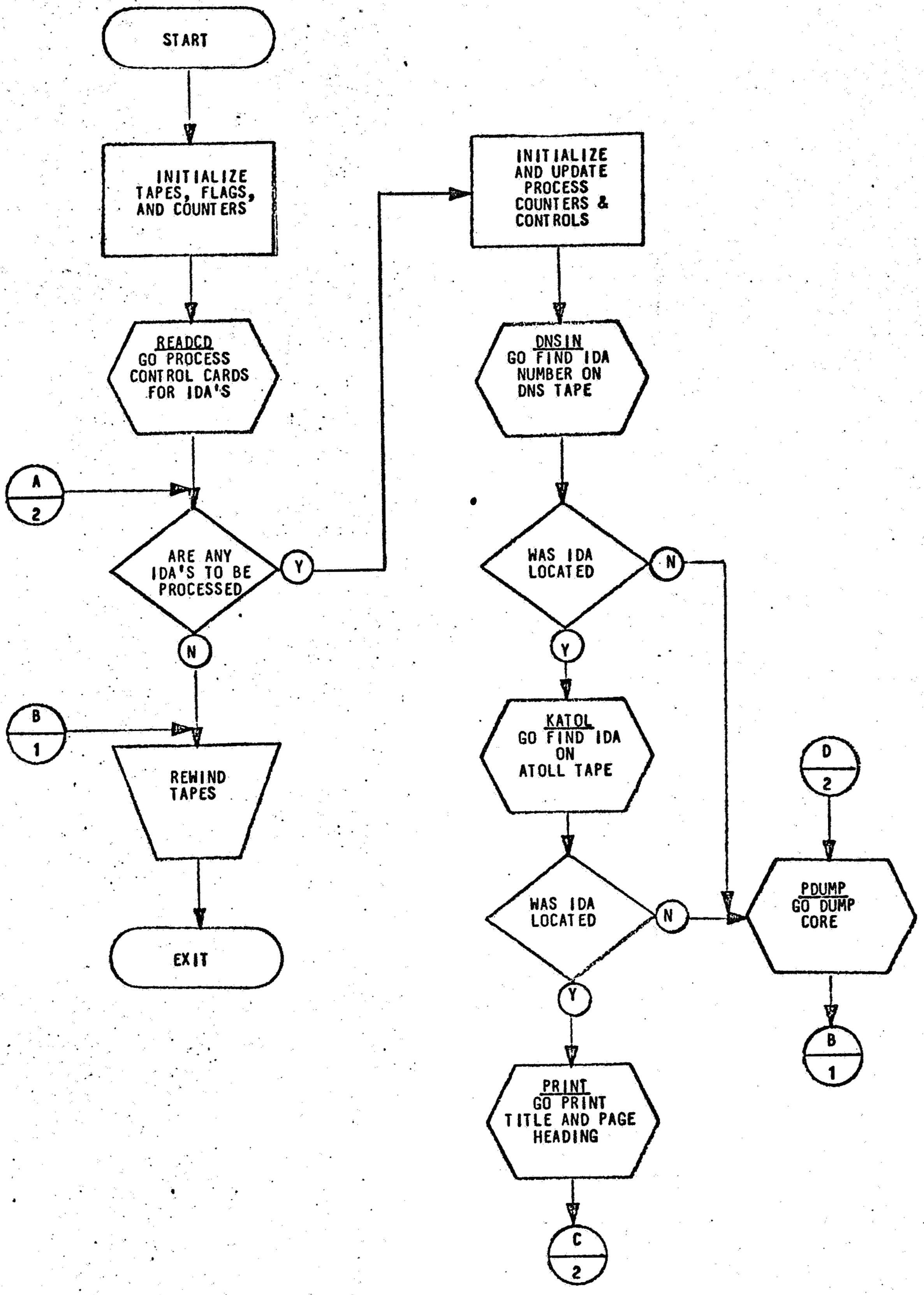
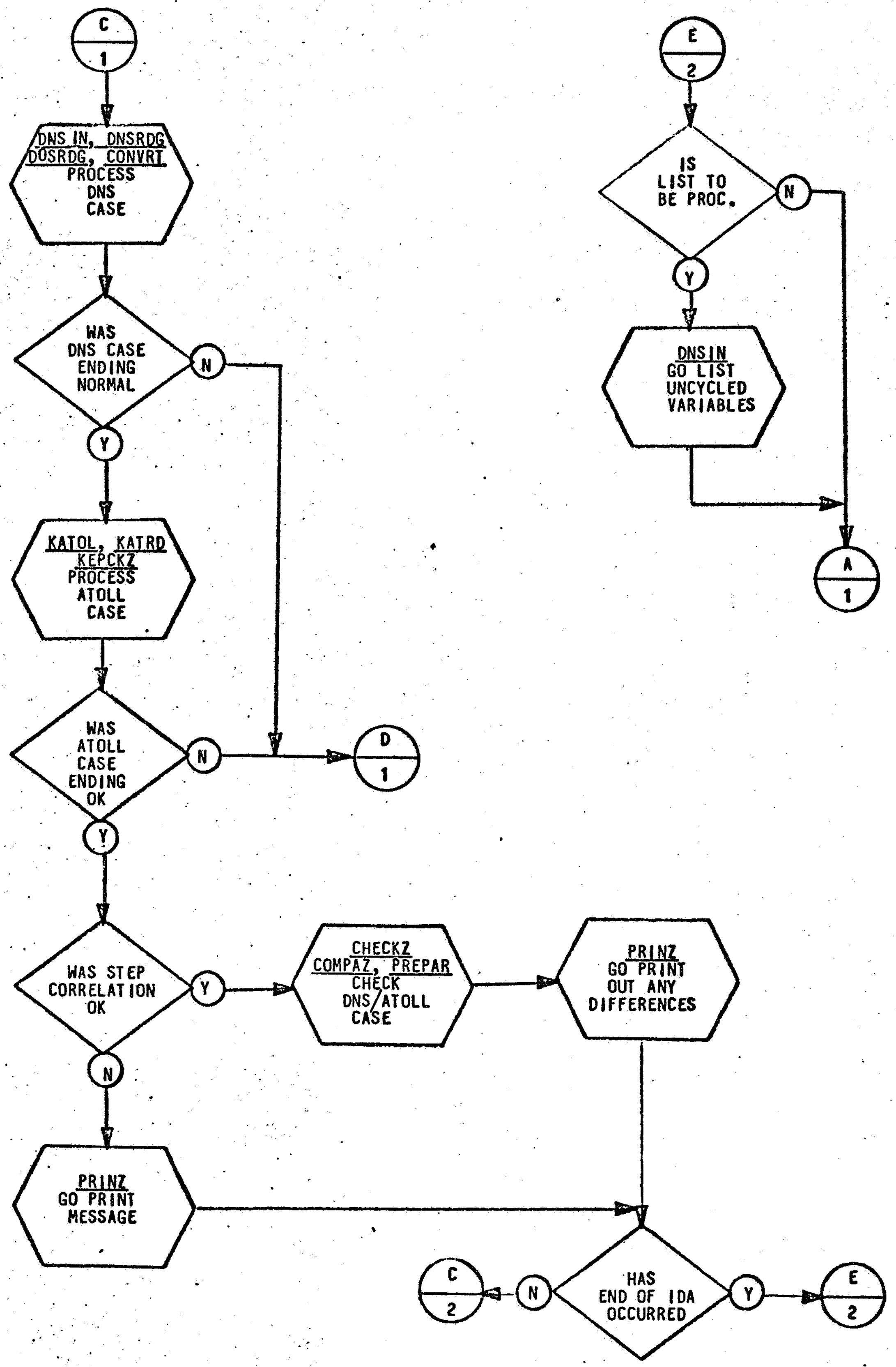

START
INITIALIZE TAPES, FLAGS, AND COUNTERS
READCD GO PROCESS CONTROL CARDS FOR IDA'S
A 2
ARE ANY IDA'S TO BE PROCESSED
Y
N
B 1
REWIND TAPES
EXIT
INITIALIZE AND UPDATE PROCESS COUNTERS & CONTROLS
DNSIN GO FIND IDA NUMBER ON DNS TAPE
WAS IDA LOCATED
N
Y
KATOL GO FIND IDA ON ATOLL TAPE
WAS IDA LOCATED
N
Y
PRINT GO PRINT TITLE AND PAGE HEADING
C 2
D 2
PDUMP GO DUMP CORE
B 1

Figure A-1. SUBROUTINE CONTRD (1 of 2)



C 1
DNS IN, DNSRDG DOSRDG, CONVRT PROCESS DNS CASE
WAS DNS CASE ENDING NORMAL
N
Y
KATOL, KATRD KEPCKZ PROCESS ATOLL CASE
WAS ATOLL CASE ENDING OK
N
D 1
Y
WAS STEP CORRELATION OK
Y
CHECKZ COMPAZ, PREPAR CHECK DNS/ATOLL CASE
PRINZ GO PRINT OUT ANY DIFFERENCES
N
PRINZ GO PRINT MESSAGE
HAS END OF IDA OCCURRED
N
C 2
Y
E 2
E 2
IS LIST TO BE PROC.
N
Y
DNSIN GO LIST UNCYCLED VARIABLES
A 1

Figure A-1. SUBROUTINE CONTRD (2 of 2)

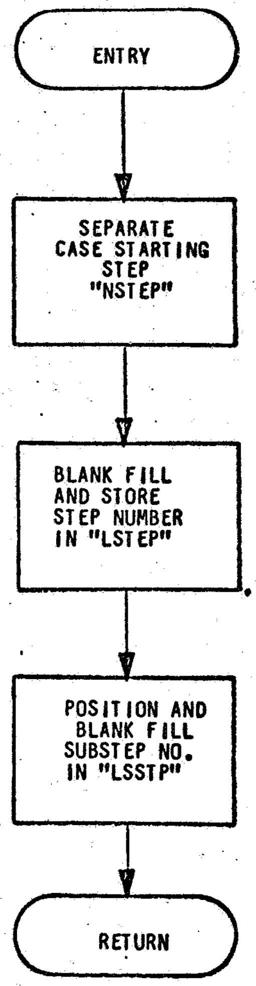

ENTRY
SEPARATE CASE STARTING STEP "NSTEP"
BLANK FILL AND STORE STEP NUMBER IN "LSTEP"
POSITION AND BLANK FILL SUBSTEP NO. IN "LSSTP"
RETURN

Figure A-2. SUBROUTINE CONVRT (1 of 1)

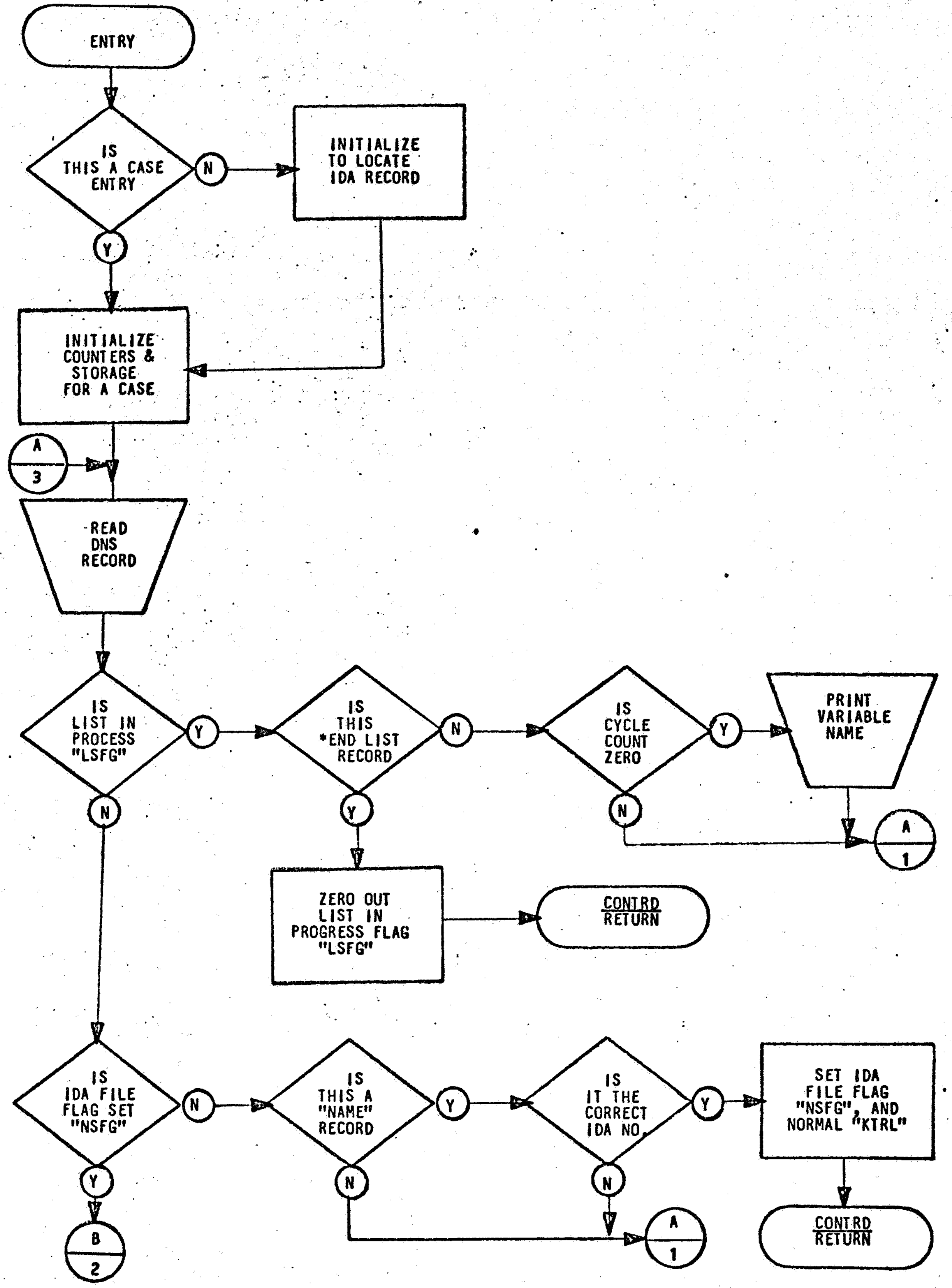

ENTRY
IS THIS A CASE ENTRY
N
INITIALIZE TO LOCATE IDA RECORD
Y
INITIALIZE COUNTERS & STORAGE FOR A CASE
A 3
READ DNS RECORD
IS LIST IN PROCESS "LSFG"
Y
IS THIS *END LIST RECORD
N
IS CYCLE COUNT ZERO
Y
PRINT VARIABLE NAME
N
A 1
Y
ZERO OUT LIST IN PROGRESS FLAG "LSFG"
CONTRD RETURN
N
IS IDA FILE FLAG SET "NSFG"
N
IS THIS A "NAME" RECORD
Y
IS IT THE CORRECT IDA NO.
Y
SET IDA FILE FLAG "NSFG", AND NORMAL "KTRL"
CONTRD RETURN
Y
B 2
N
N
A 1

Figure A-3. SUBROUTINE DNSIN (1 of 3)

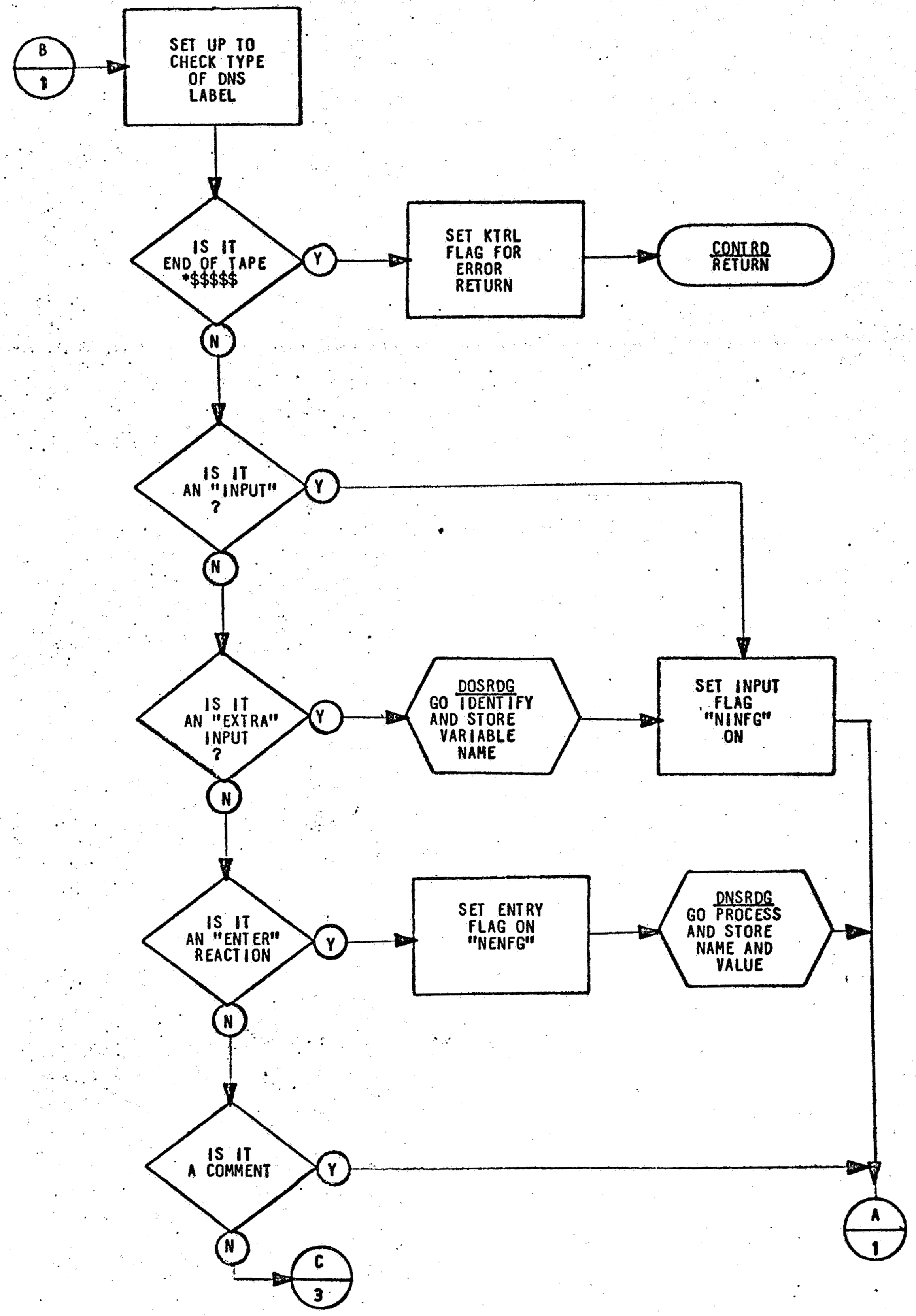

B
1
SET UP TO CHECK TYPE OF DNS LABEL
IS IT END OF TAPE *$$$$$
Y
N
SET KTRL FLAG FOR ERROR RETURN
CONTRD RETURN
IS IT AN "INPUT" ?
Y
N
SET INPUT FLAG "NINFG" ON
IS IT AN "EXTRA" INPUT ?
Y
N
DOSRDG GO IDENTIFY AND STORE VARIABLE NAME
IS IT AN "ENTER" REACTION
Y
N
SET ENTRY FLAG ON "NENFG"
DNSRDG GO PROCESS AND STORE NAME AND VALUE
IS IT A COMMENT
Y
N
C
3
A
1

Figure A-3. SUBROUTINE DNSIN (2 of 3)

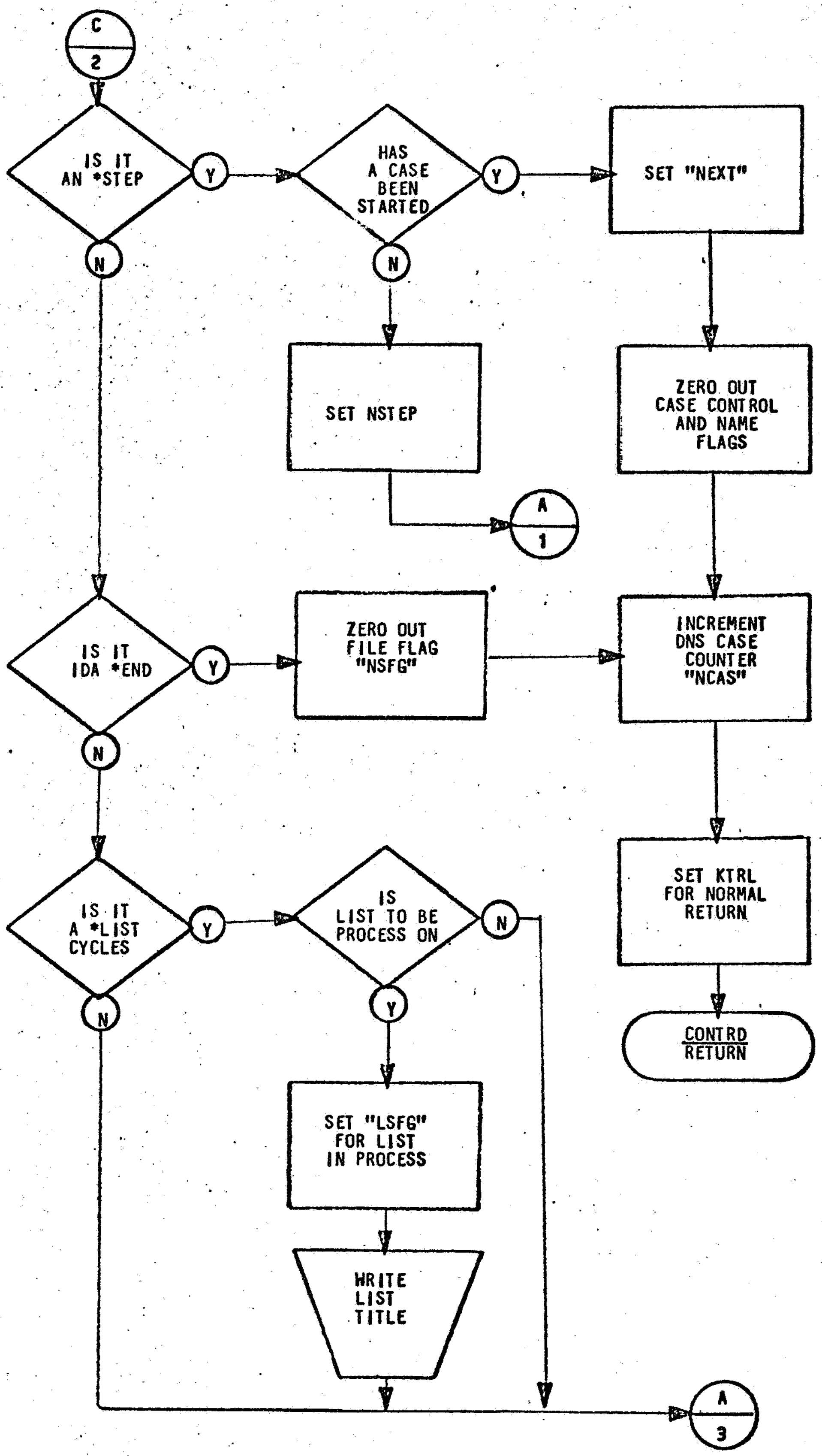


Figure A-3. SUBROUTINE DNSIN (3 of 3)

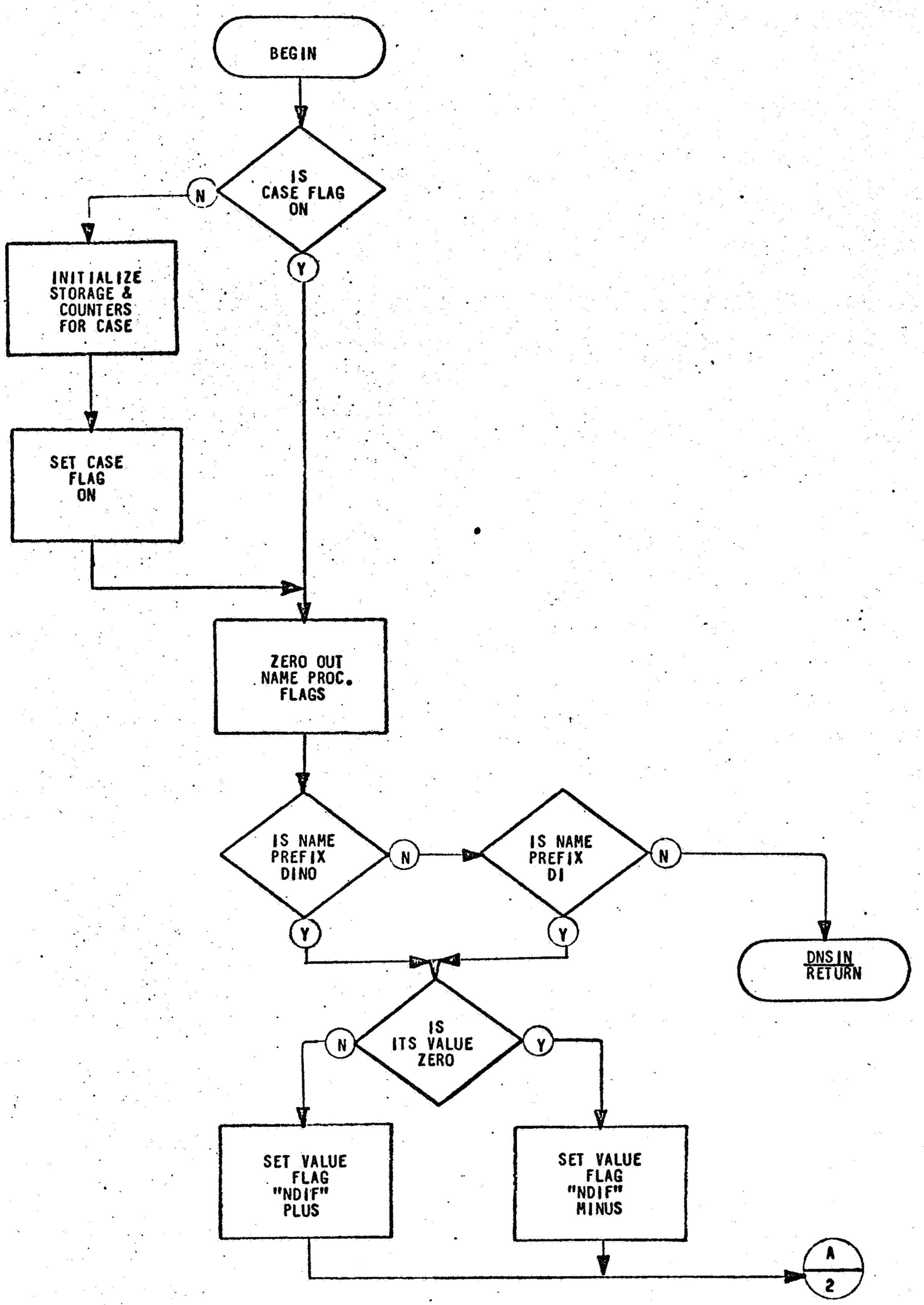

BEGIN
IS CASE FLAG ON
N
Y
INITIALIZE STORAGE & COUNTERS FOR CASE
SET CASE FLAG ON
ZERO OUT NAME PROC. FLAGS
IS NAME PREFIX DINO
N
Y
IS NAME PREFIX DI
N
Y
DNSIN RETURN
IS ITS VALUE ZERO
N
Y
SET VALUE FLAG "NDIF" PLUS
SET VALUE FLAG "NDIF" MINUS
A
2

Figure A-4. SUBROUTINE DNSRDG (1 of 2)

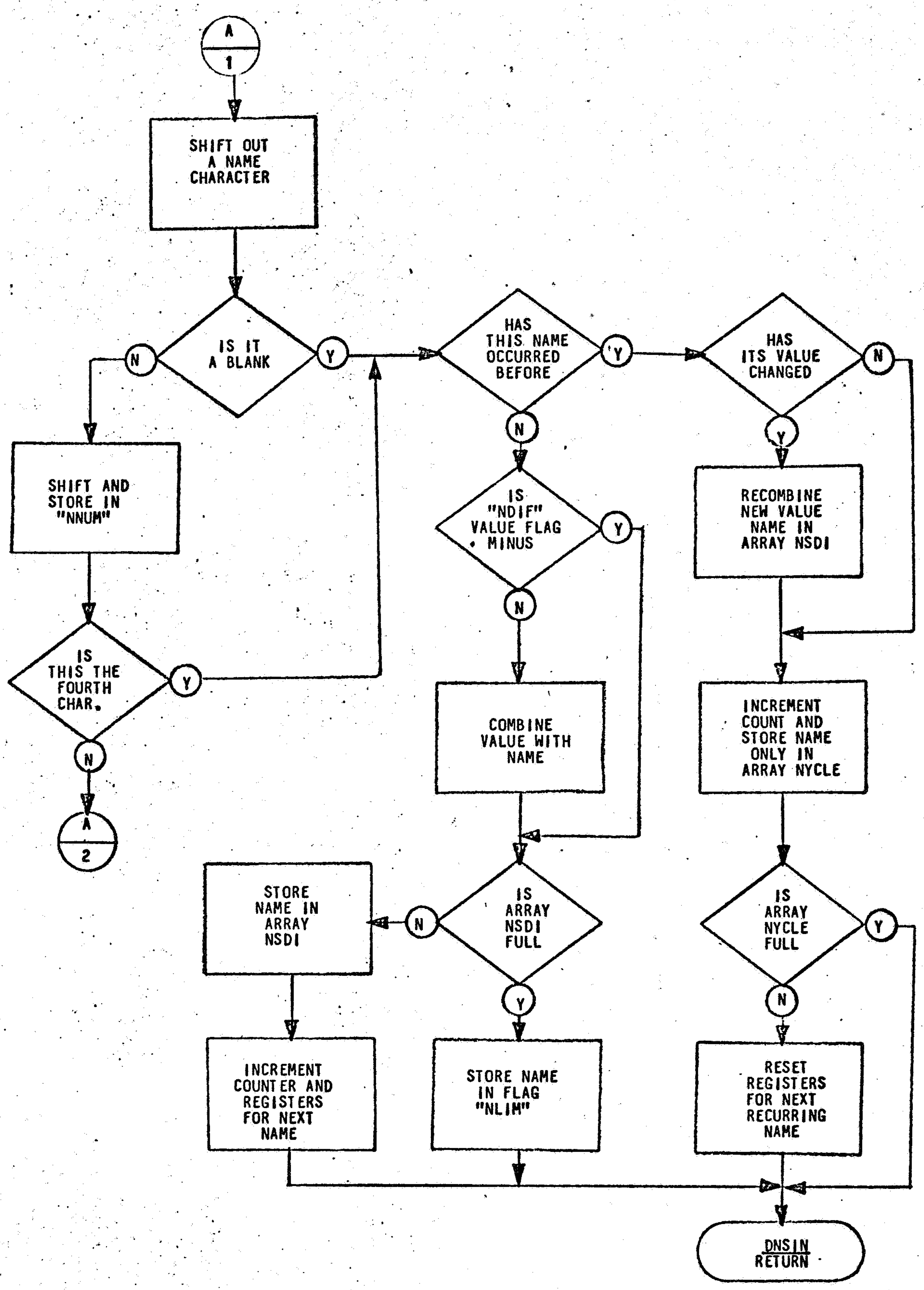

A
1
SHIFT OUT A NAME CHARACTER
IS IT A BLANK
N
Y
SHIFT AND STORE IN "NNUM"
IS THIS THE FOURTH CHAR.
Y
N
A
2
HAS THIS NAME OCCURRED BEFORE
Y
N
IS "NDIF" VALUE FLAG MINUS
Y
N
COMBINE VALUE WITH NAME
IS ARRAY NSDI FULL
N
Y
STORE NAME IN ARRAY NSDI
INCREMENT COUNTER AND REGISTERS FOR NEXT NAME
STORE NAME IN FLAG "NLIM"
HAS ITS VALUE CHANGED
N
Y
RECOMBINE NEW VALUE NAME IN ARRAY NSDI
INCREMENT COUNT AND STORE NAME ONLY IN ARRAY NYCLE
IS ARRAY NYCLE FULL
Y
N
RESET REGISTERS FOR NEXT RECURRING NAME
DNSIN RETURN

Figure A-4. SUBROUTINE DNSRDG (2 of 2)

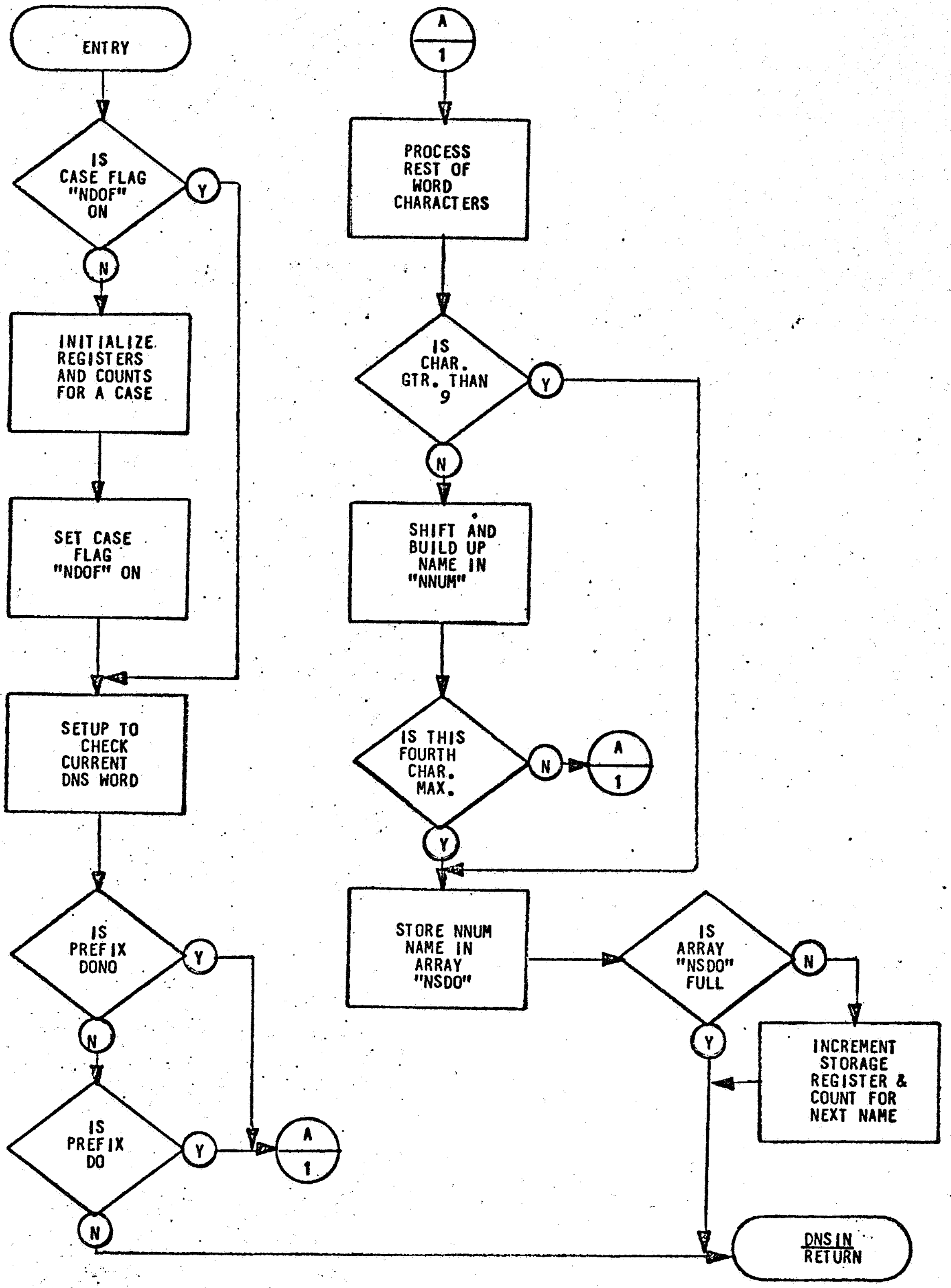

ENTRY
IS CASE FLAG "NDOF" ON
Y
N
INITIALIZE REGISTERS AND COUNTS FOR A CASE
SET CASE FLAG "NDOF" ON
SETUP TO CHECK CURRENT DNS WORD
IS PREFIX DONO
Y
N
IS PREFIX DO
Y
N
A 1
A 1
PROCESS REST OF WORD CHARACTERS
IS CHAR. GTR. THAN 9
Y
N
SHIFT AND BUILD UP NAME IN "NNUM"
IS THIS FOURTH CHAR. MAX.
N
A 1
Y
STORE NNUM NAME IN ARRAY "NSDO"
IS ARRAY "NSDO" FULL
N
Y
INCREMENT STORAGE REGISTER & COUNT FOR NEXT NAME
DNSIN RETURN

Figure A-5. SUBROUTINE DOSRDG (1 of 1)

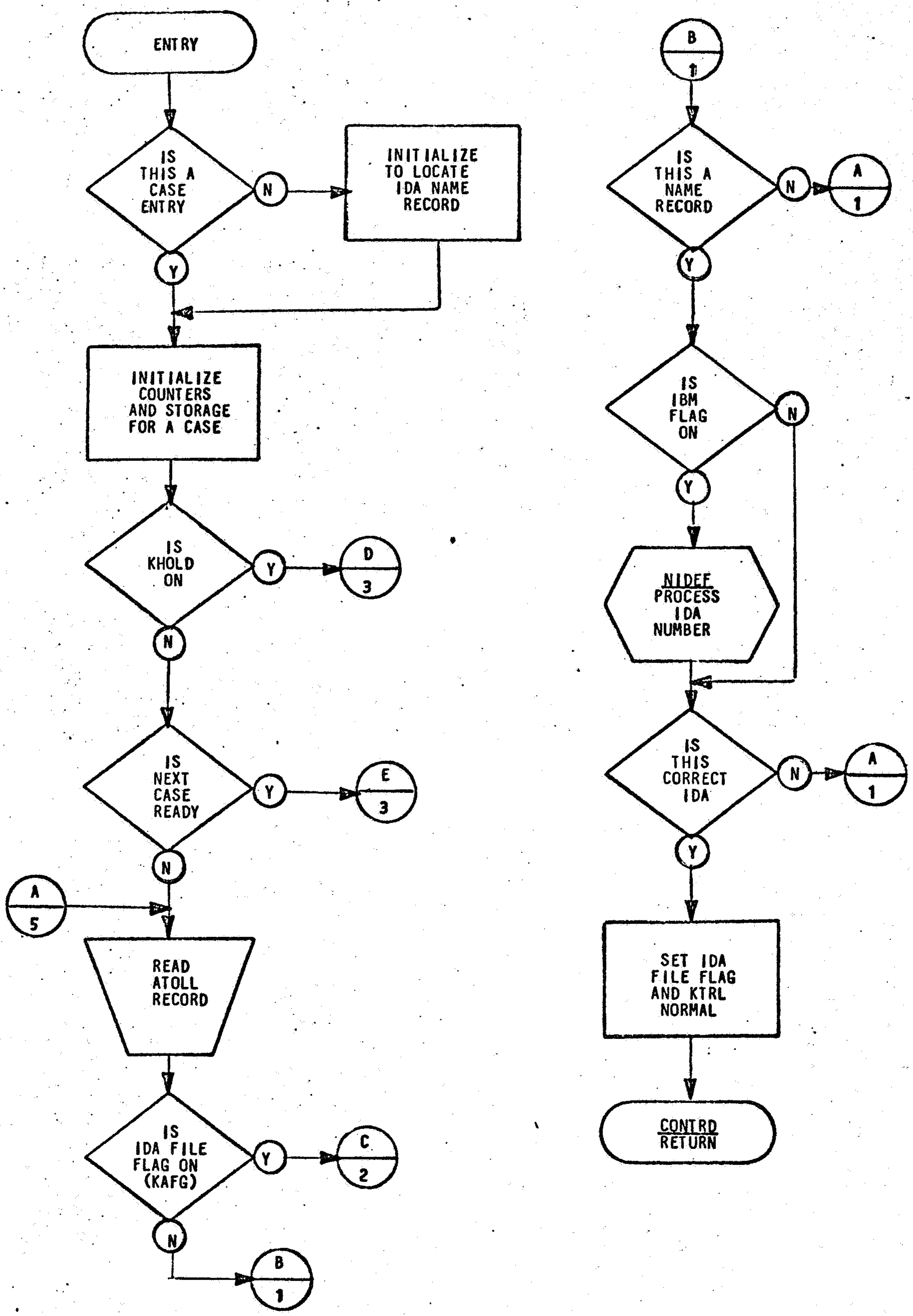


Figure A-6. SUBROUTINE KATOL (1 of 5)

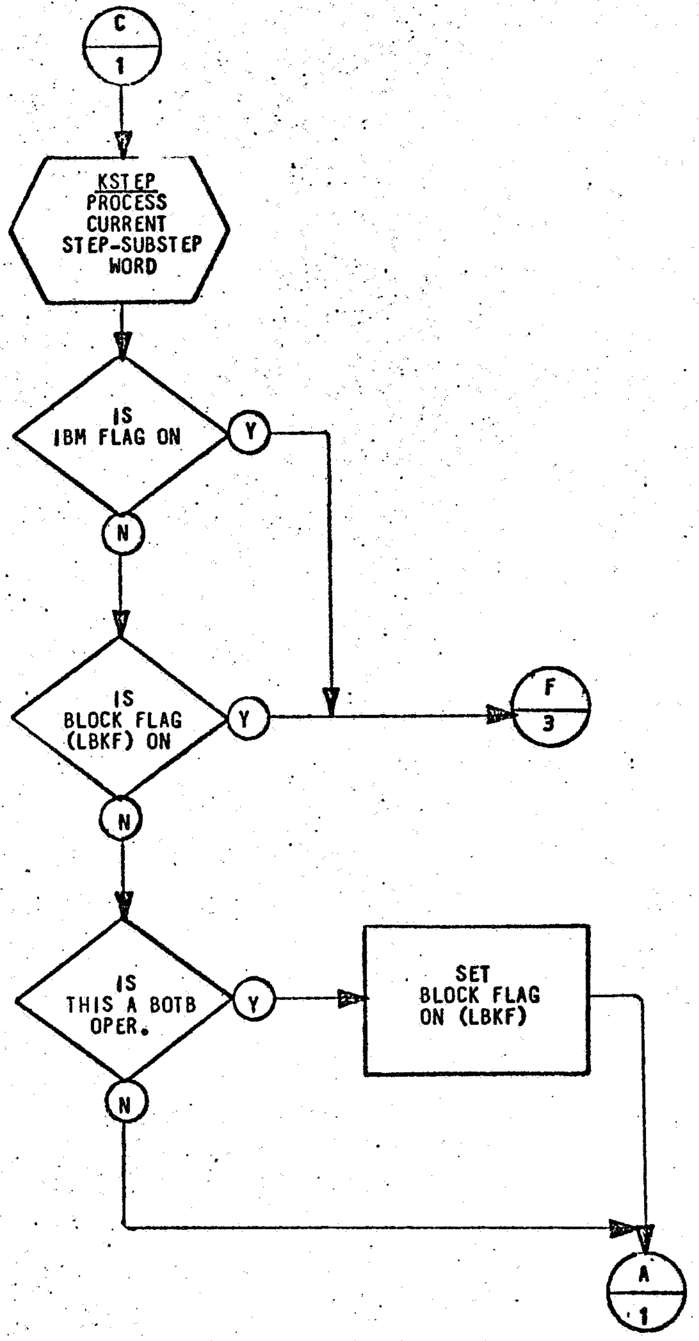


Figure A-6. SUBROUTINE KATOL (2 of 5)

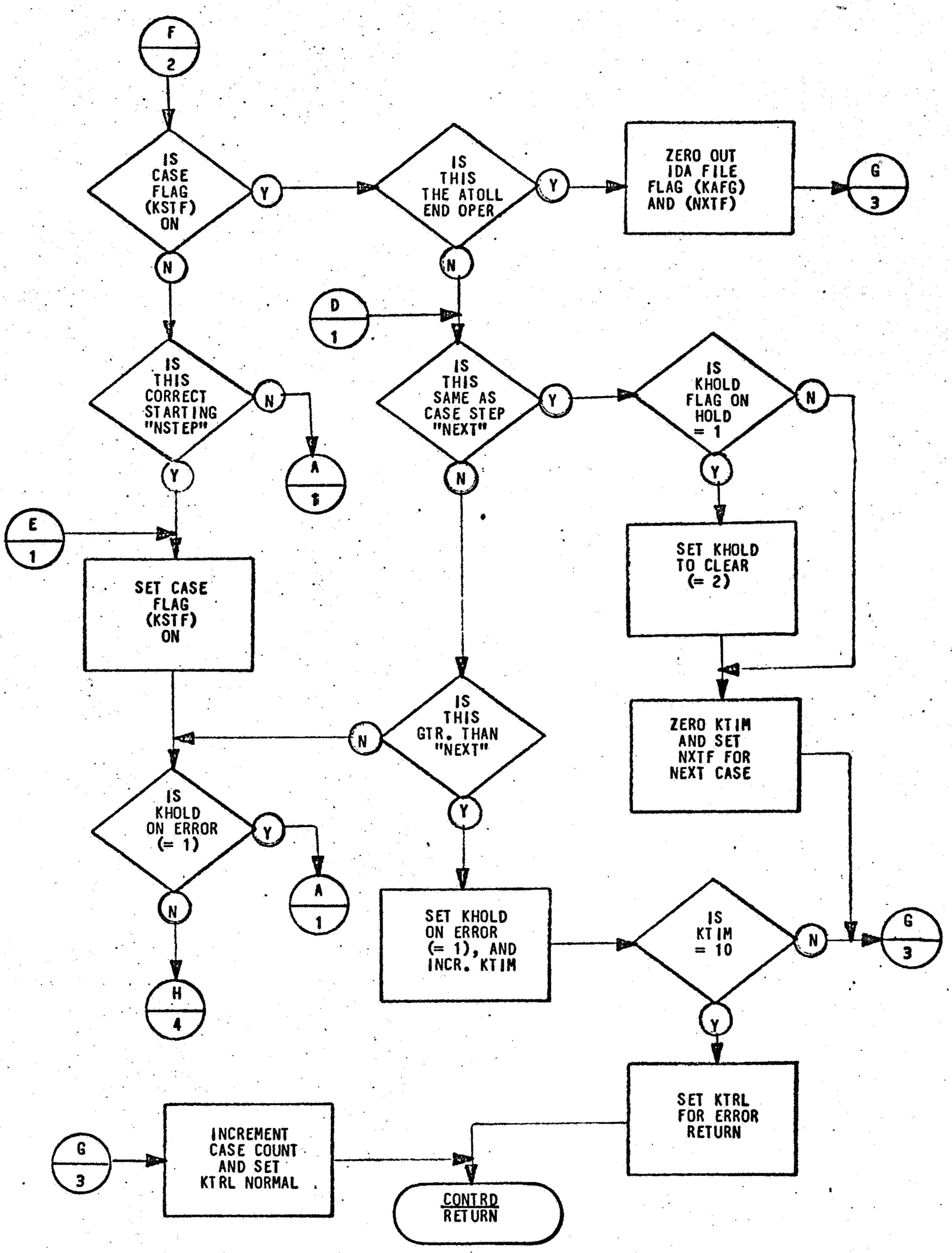

F 2
IS CASE FLAG (KSTF) ON
Y
N
IS THIS THE ATOLL END OPER
ZERO OUT IDA FILE FLAG (KAFG) AND (NXTF)
G 3
D 1
IS THIS CORRECT STARTING "NSTEP"
A 1
IS THIS SAME AS CASE STEP "NEXT"
IS KHOLD FLAG ON HOLD = 1
E 1
SET CASE FLAG (KSTF) ON
SET KHOLD TO CLEAR (= 2)
ZERO KTIM AND SET NXTF FOR NEXT CASE
IS THIS GTR. THAN "NEXT"
IS KHOLD ON ERROR (= 1)
A 1
SET KHOLD ON ERROR (= 1), AND INCR. KTIM
IS KTIM = 10
G 3
H 4
SET KTRL FOR ERROR RETURN
G 3
INCREMENT CASE COUNT AND SET KTRL NORMAL
CONTRD RETURN

Figure A-6. SUBROUTINE KATOL (3 of 5)

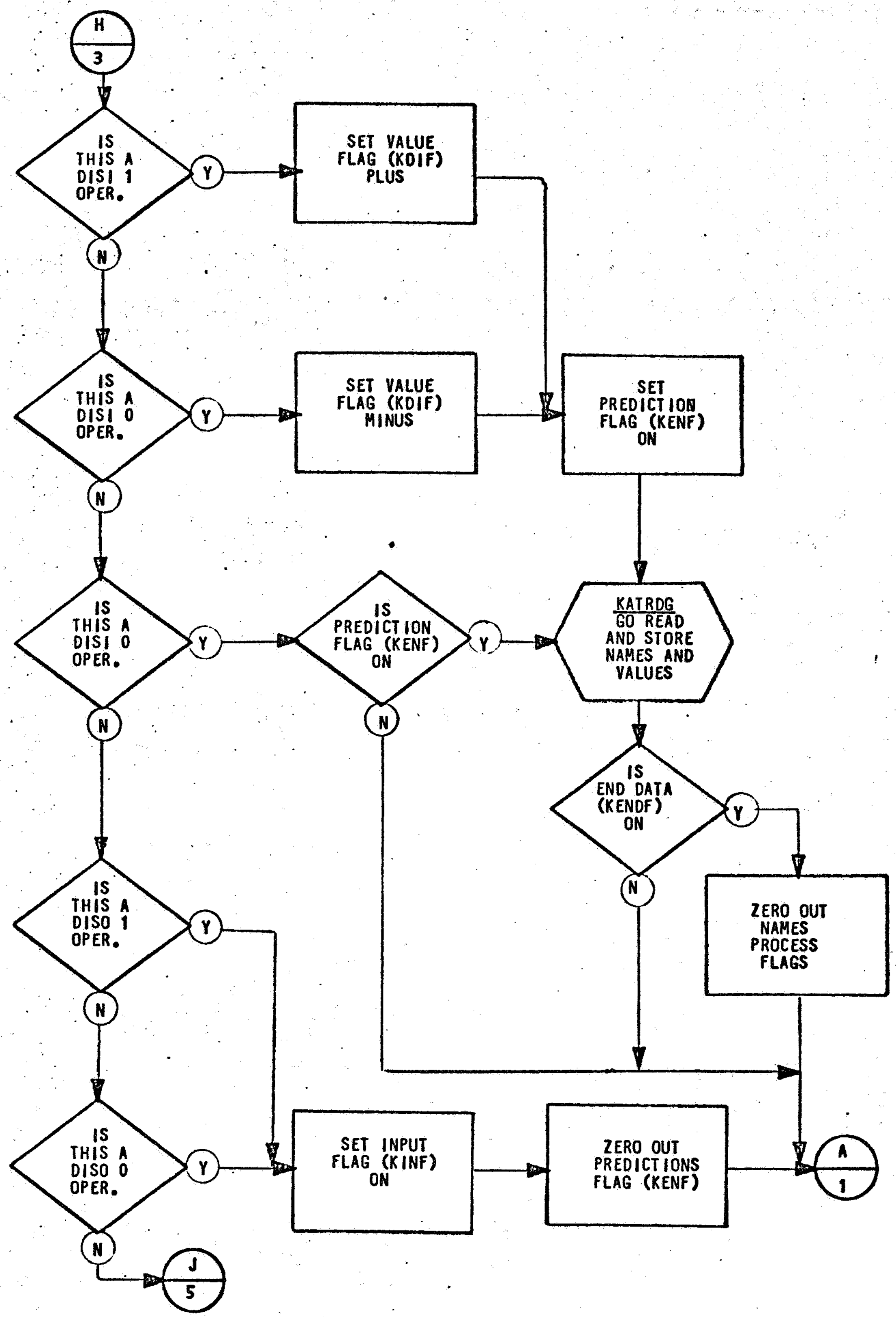

H 3
IS THIS A DISI 1 OPER.
Y
N
SET VALUE FLAG (KDIF) PLUS
IS THIS A DISI 0 OPER.
Y
N
SET VALUE FLAG (KDIF) MINUS
SET PREDICTION FLAG (KENF) ON
IS THIS A DISI 0 OPER.
Y
N
IS PREDICTION FLAG (KENF) ON
Y
N
KATRDG GO READ AND STORE NAMES AND VALUES
IS END DATA (KENDF) ON
Y
N
ZERO OUT NAMES PROCESS FLAGS
IS THIS A DISO 1 OPER.
Y
N
IS THIS A DISO 0 OPER.
Y
N
J 5
SET INPUT FLAG (KINF) ON
ZERO OUT PREDICTIONS FLAG (KENF)
A 1

Figure A-6. SUBROUTINE KATOL (4 of 5)

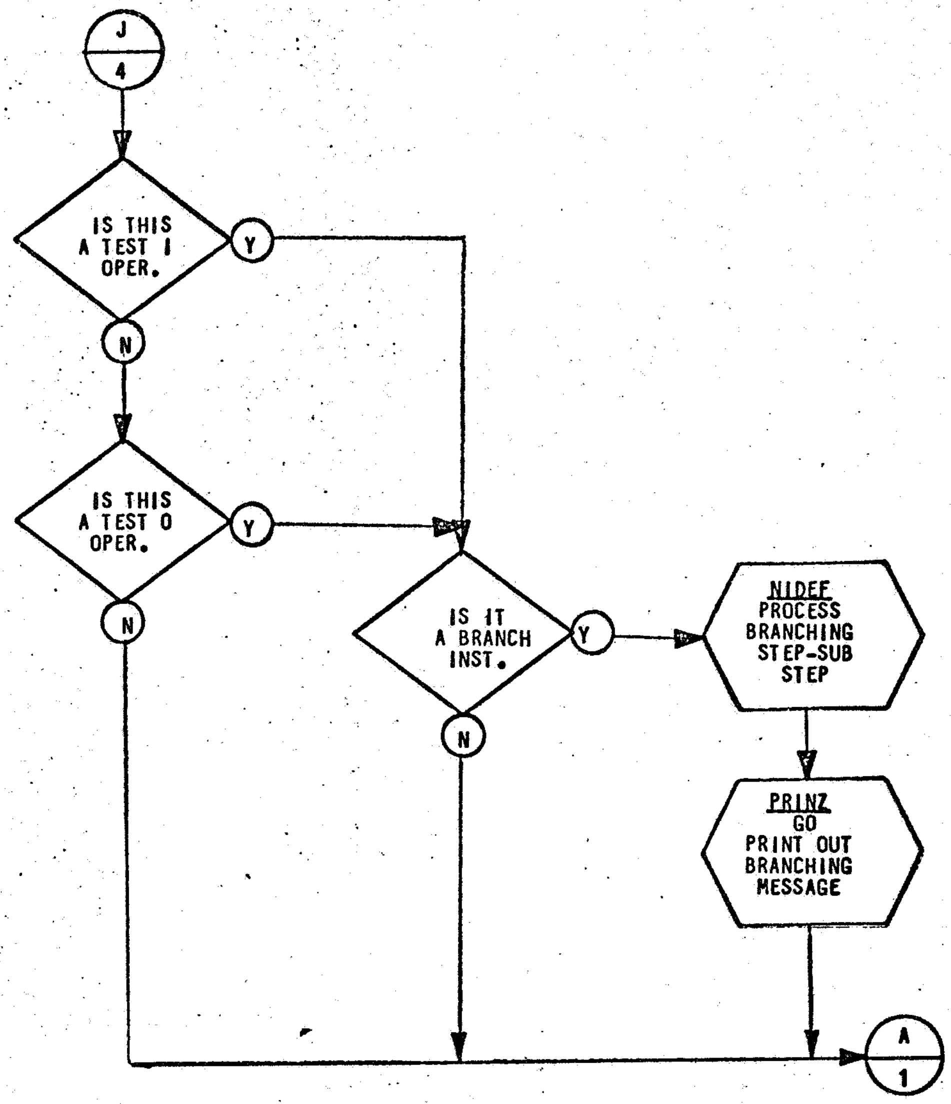


Figure A-6. SUBROUTINE KATOL (5 of 5)

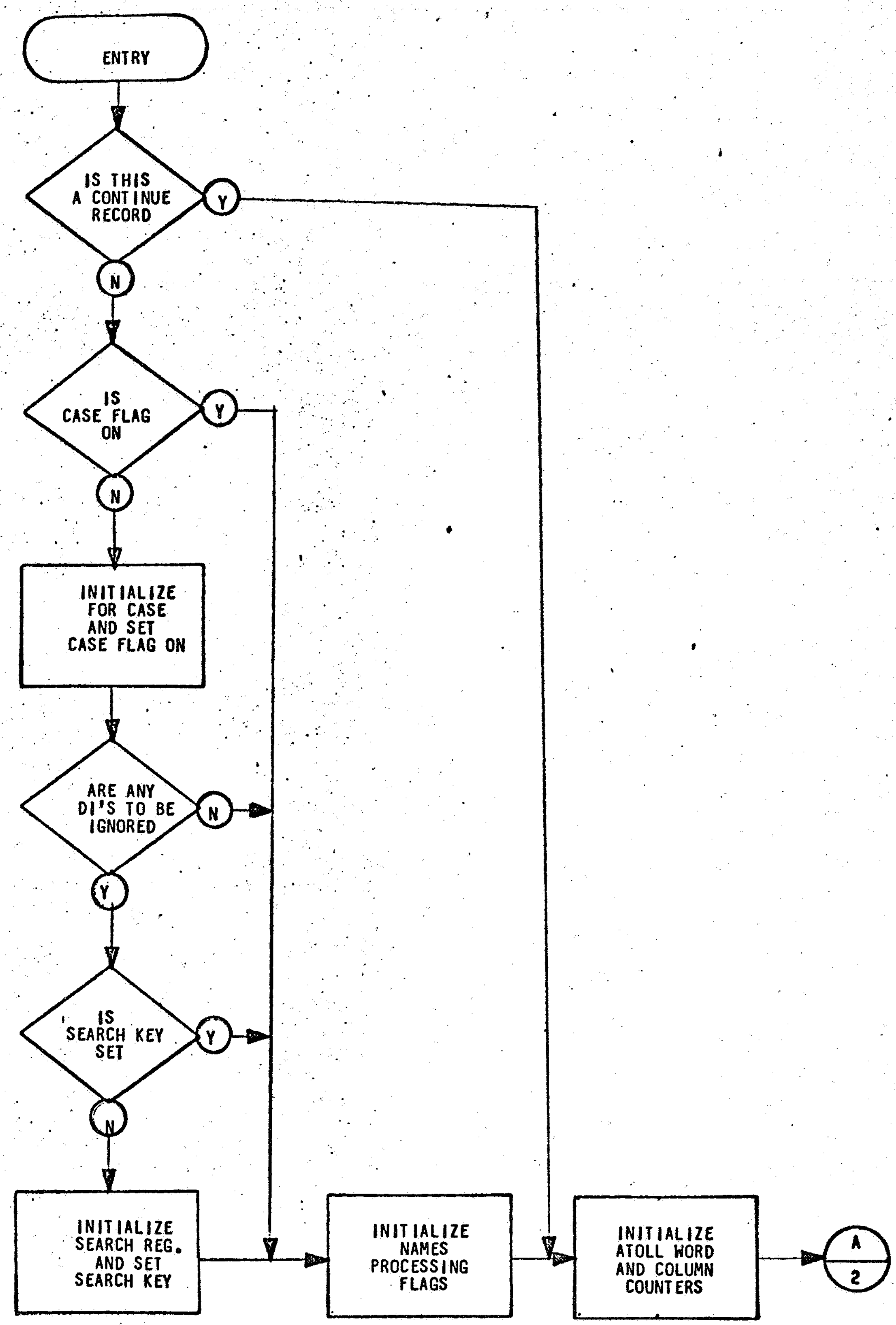

ENTRY
IS THIS A CONTINUE RECORD
Y
N
IS CASE FLAG ON
Y
N
INITIALIZE FOR CASE AND SET CASE FLAG ON
ARE ANY DI'S TO BE IGNORED
N
Y
IS SEARCH KEY SET
Y
N
INITIALIZE SEARCH REG. AND SET SEARCH KEY
INITIALIZE NAMES PROCESSING FLAGS
INITIALIZE ATOLL WORD AND COLUMN COUNTERS
A
2

Figure A-7. SUBROUTINE KATRDG (1 of 5)

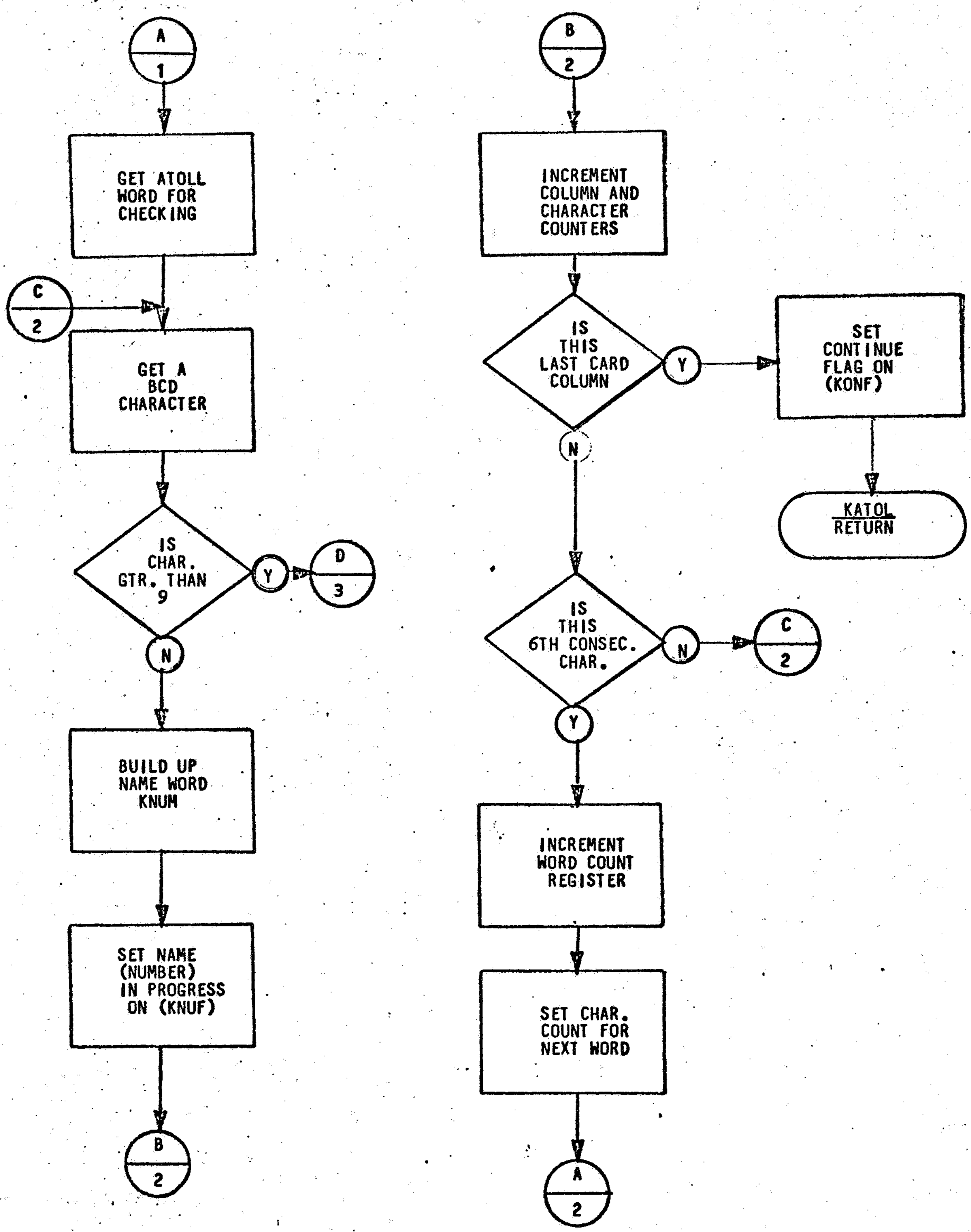

A
1
GET ATOLL WORD FOR CHECKING
C
2
GET A BCD CHARACTER
IS CHAR. GTR. THAN 9
Y
D
3
N
BUILD UP NAME WORD KNUM
SET NAME (NUMBER) IN PROGRESS ON (KNUF)
B
2
B
2
INCREMENT COLUMN AND CHARACTER COUNTERS
IS THIS LAST CARD COLUMN
Y
SET CONTINUE FLAG ON (KONF)
KATOL RETURN
N
IS THIS 6TH CONSEC. CHAR.
N
C
2
Y
INCREMENT WORD COUNT REGISTER
SET CHAR. COUNT FOR NEXT WORD
A
2

Figure A-7. SUBROUTINE KATRDG (2 of 5)

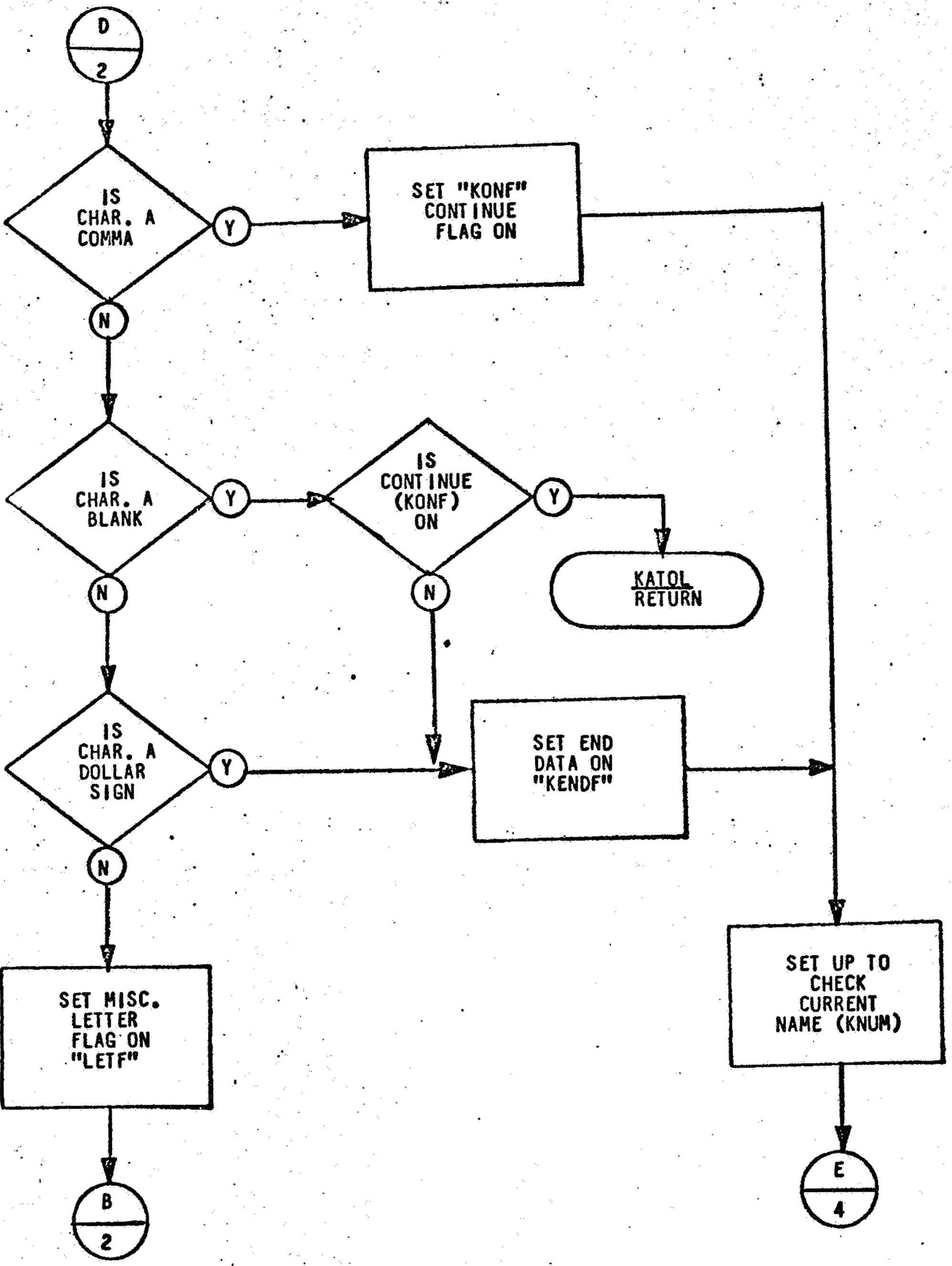


Figure A-7. SUBROUTINE KATRDG (3 of 5)

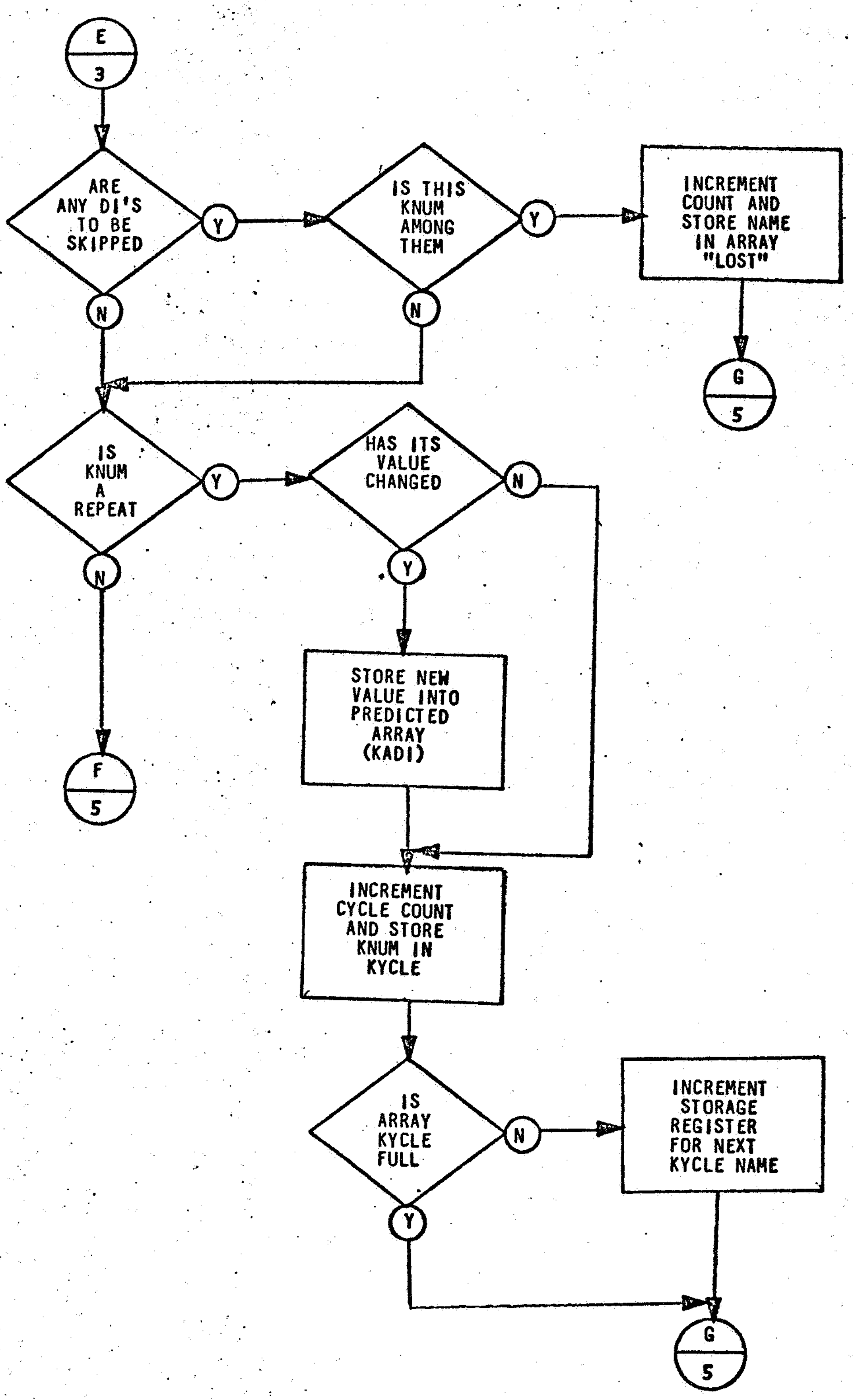

E
3
ARE ANY DI'S TO BE SKIPPED
Y
N
IS THIS KNUM AMONG THEM
Y
N
INCREMENT COUNT AND STORE NAME IN ARRAY "LOST"
G
5
IS KNUM A REPEAT
Y
N
HAS ITS VALUE CHANGED
N
Y
F
5
STORE NEW VALUE INTO PREDICTED ARRAY (KADI)
INCREMENT CYCLE COUNT AND STORE KNUM IN KYCLE
IS ARRAY KYCLE FULL
N
Y
INCREMENT STORAGE REGISTER FOR NEXT KYCLE NAME
G
5

Figure A-7. SUBROUTINE KATRDG (4 of 5)

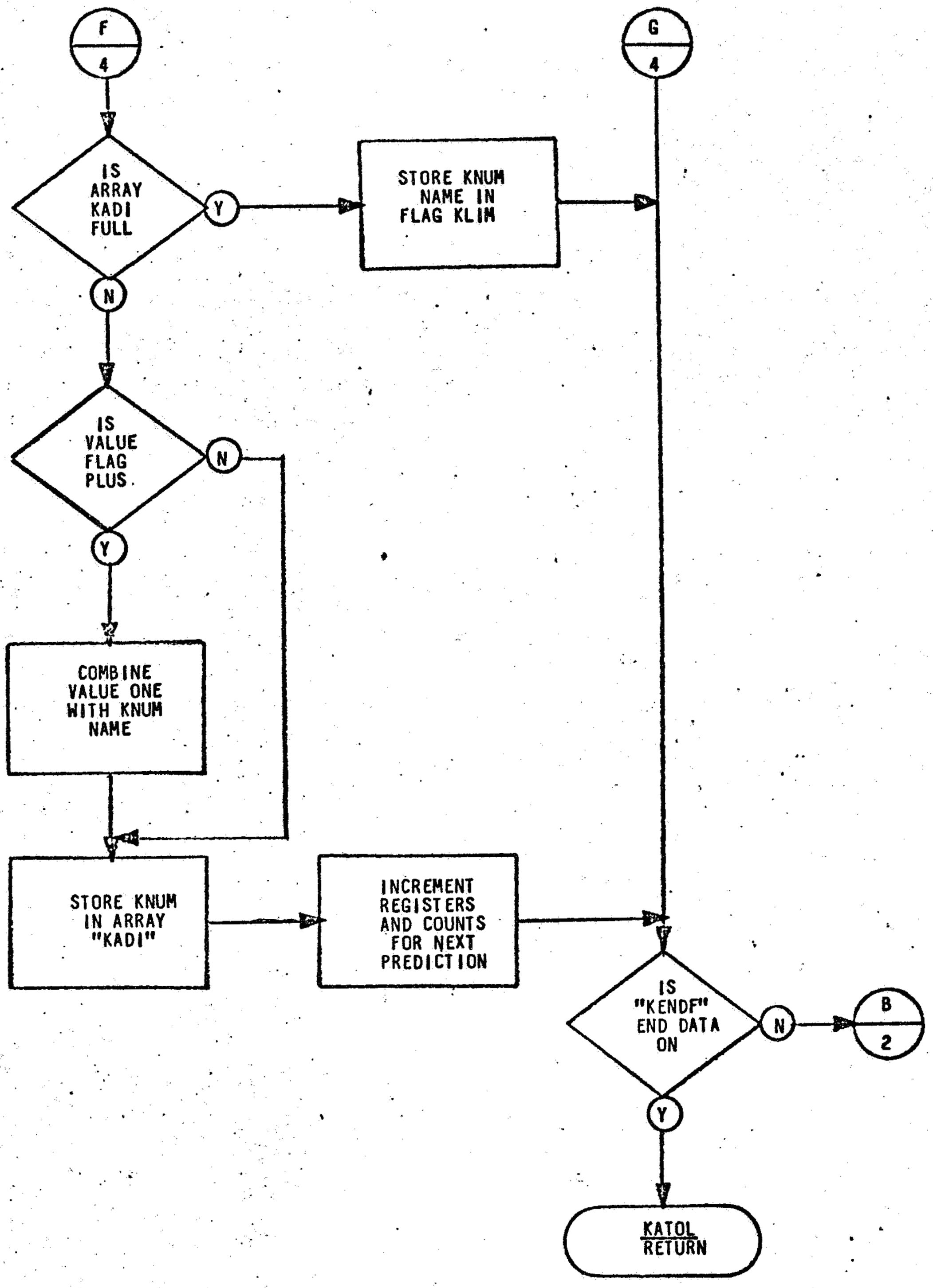

F
4
G
4
IS ARRAY KADI FULL
Y
N
STORE KNUM NAME IN FLAG KLIM
IS VALUE FLAG PLUS
N
Y
COMBINE VALUE ONE WITH KNUM NAME
STORE KNUM IN ARRAY "KADI"
INCREMENT REGISTERS AND COUNTS FOR NEXT PREDICTION
IS "KENDF" END DATA ON
N
B
2
Y
KATOL RETURN

Figure A-7. SUBROUTINE KATRDG (5 of 5)

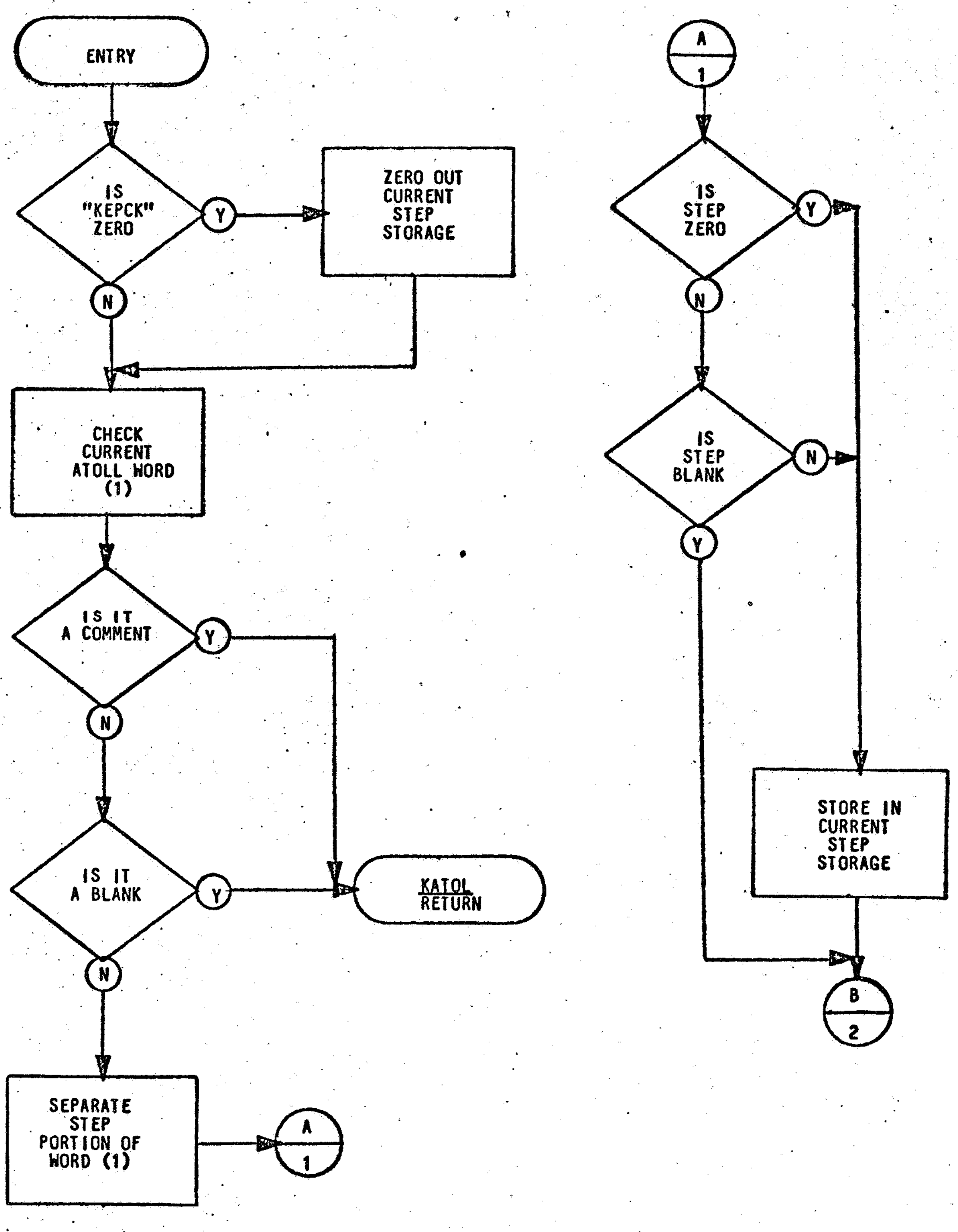

ENTRY
IS "KEPCK" ZERO
Y
N
ZERO OUT CURRENT STEP STORAGE
CHECK CURRENT ATOLL WORD (1)
IS IT A COMMENT
Y
N
IS IT A BLANK
Y
N
KATOL RETURN
SEPARATE STEP PORTION OF WORD (1)
A
1
A
1
IS STEP ZERO
Y
N
IS STEP BLANK
N
Y
STORE IN CURRENT STEP STORAGE
B
2

Figure A-8. SUBROUTINE KEPCKZ (1 of 2)

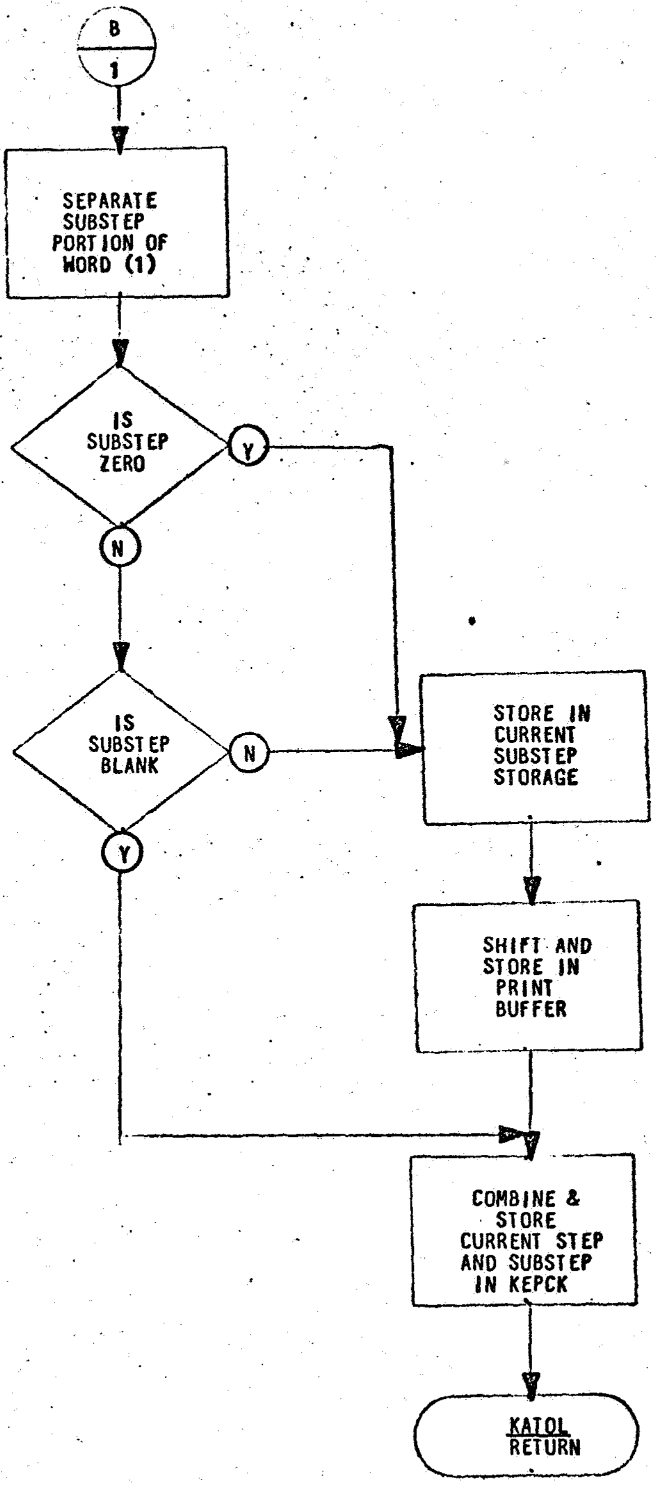

B
1
SEPARATE SUBSTEP PORTION OF WORD (1)
IS SUBSTEP ZERO
Y
N
IS SUBSTEP BLANK
N
Y
STORE IN CURRENT SUBSTEP STORAGE
SHIFT AND STORE IN PRINT BUFFER
COMBINE & STORE CURRENT STEP AND SUBSTEP IN KEPCK
KATOL RETURN

Figure A-8. SUBROUTINE KEPCKZ (2 of 2)

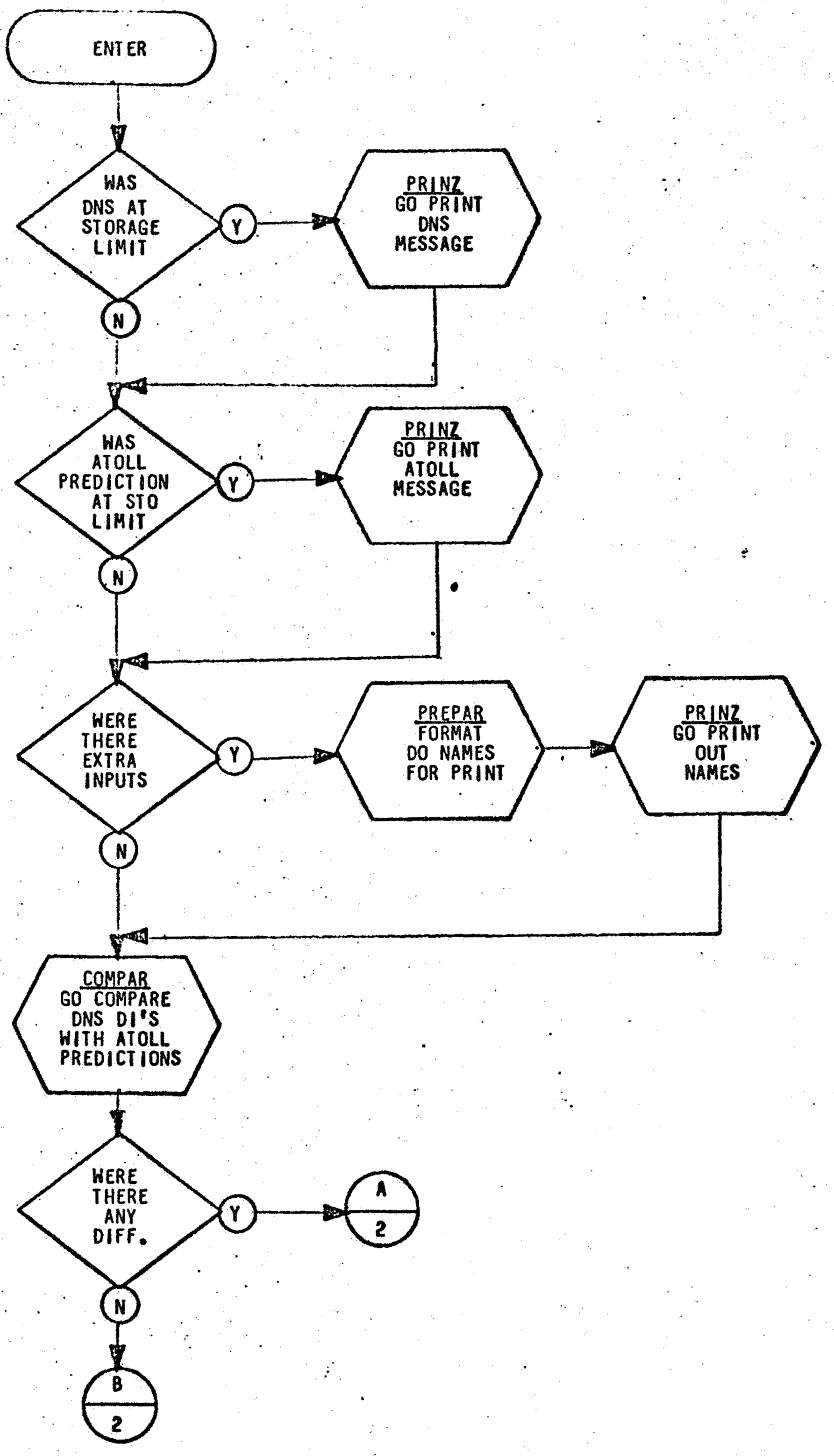

ENTER
WAS DNS AT STORAGE LIMIT
Y
N
PRINZ GO PRINT DNS MESSAGE
WAS ATOLL PREDICTION AT STO LIMIT
Y
N
PRINZ GO PRINT ATOLL MESSAGE
WERE THERE EXTRA INPUTS
Y
N
PREPAR FORMAT DO NAMES FOR PRINT
PRINZ GO PRINT OUT NAMES
COMPAR GO COMPARE DNS DI'S WITH ATOLL PREDICTIONS
WERE THERE ANY DIFF.
Y
N
A 2
B 2

Figure A-9. SUBROUTINE CHEKZ (1 of 3)

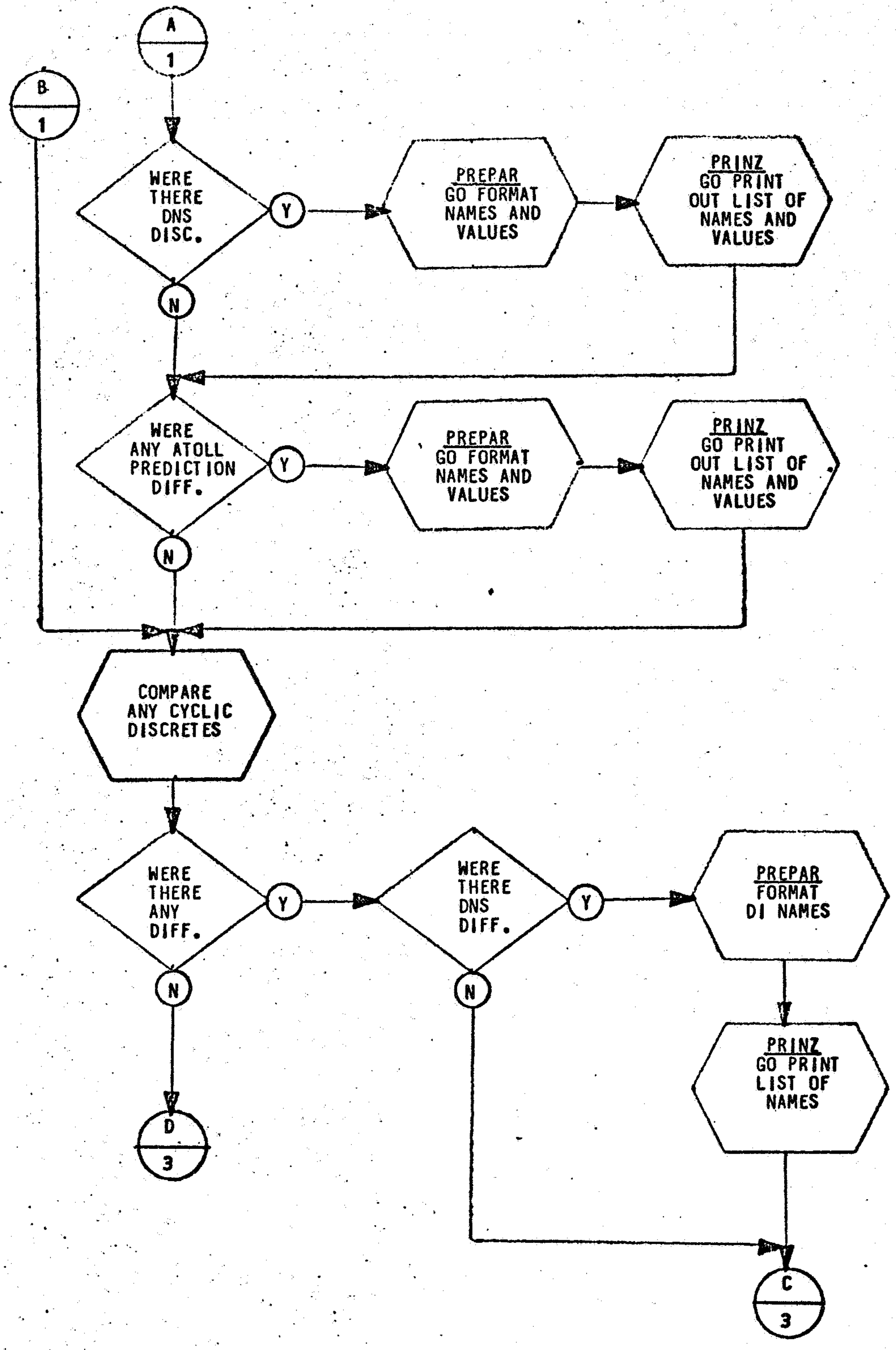
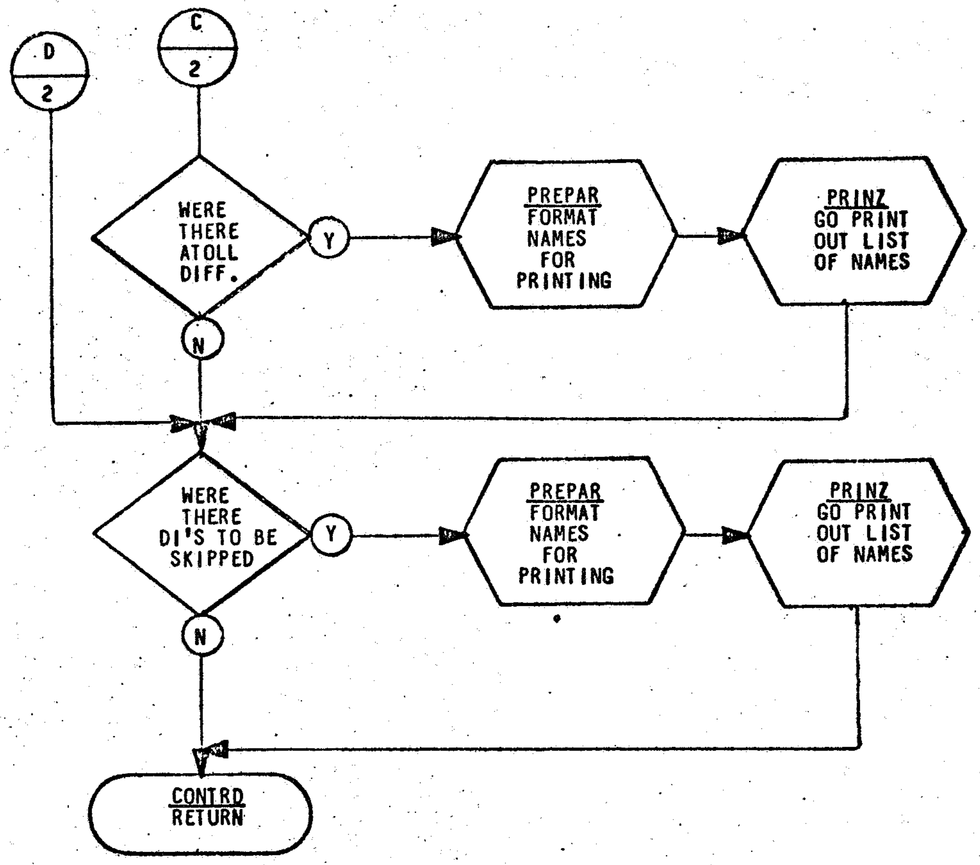

A
1
B
1
WERE THERE DNS DISC.
Y
N
PREPAR
GO FORMAT NAMES AND VALUES
PRINZ
GO PRINT OUT LIST OF NAMES AND VALUES
WERE ANY ATOLL PREDICTION DIFF.
Y
N
PREPAR
GO FORMAT NAMES AND VALUES
PRINZ
GO PRINT OUT LIST OF NAMES AND VALUES
COMPARE ANY CYCLIC DISCRETES
WERE THERE ANY DIFF.
Y
N
WERE THERE DNS DIFF.
Y
N
PREPAR
FORMAT DI NAMES
PRINZ
GO PRINT LIST OF NAMES
D
3
C
3

Figure A-9. SUBROUTINE CHEKZ (2 of 3)



D 2
C 2
WERE THERE ATOLL DIFF.
Y
N
PREPAR FORMAT NAMES FOR PRINTING
PRINZ GO PRINT OUT LIST OF NAMES
WERE THERE DI'S TO BE SKIPPED
Y
N
PREPAR FORMAT NAMES FOR PRINTING
PRINZ GO PRINT OUT LIST OF NAMES
CONTRD RETURN

Figure A-9. SUBROUTINE CHEKZ (3 of 3)

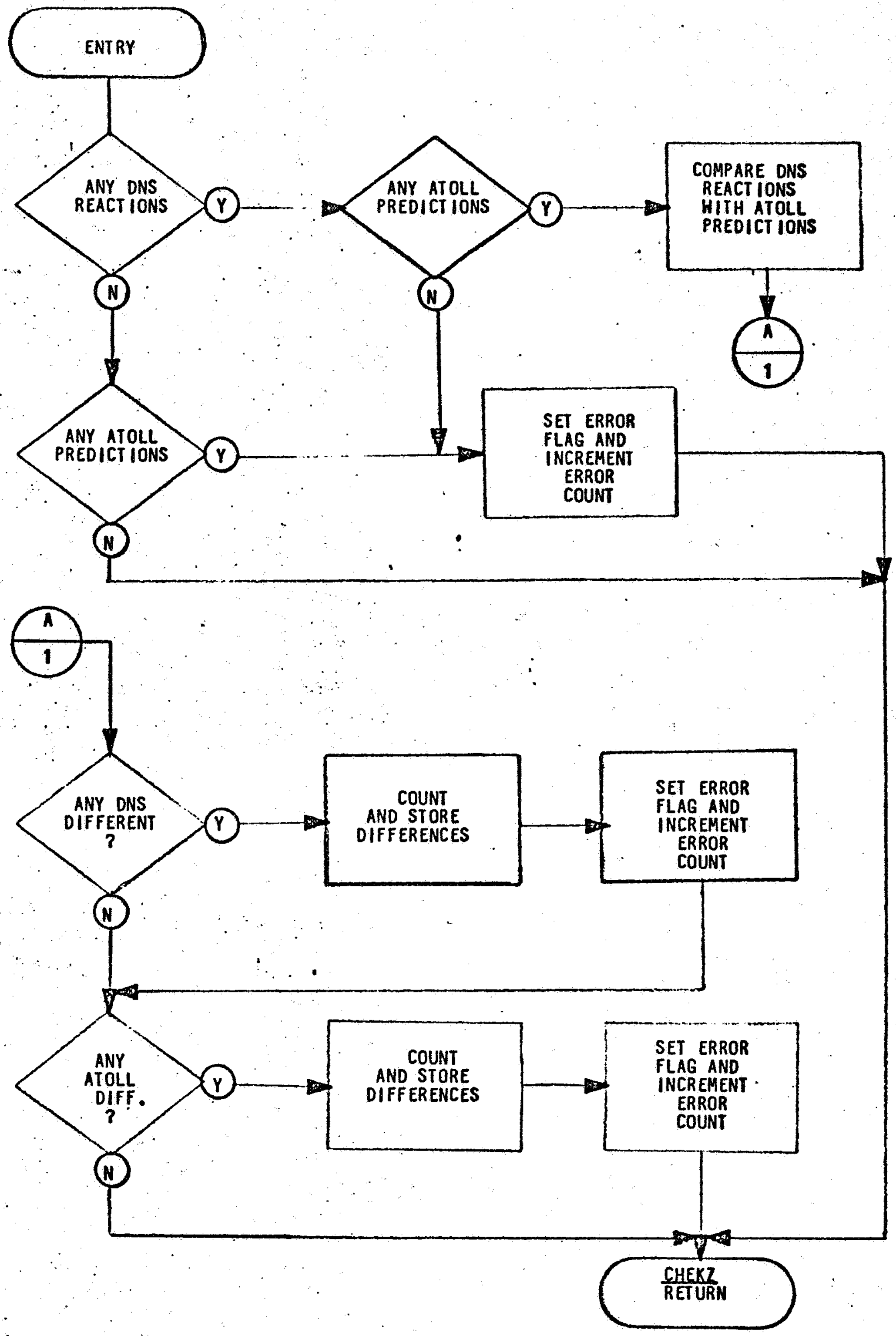

ENTRY
ANY DNS REACTIONS
Y
N
ANY ATOLL PREDICTIONS
Y
N
COMPARE DNS REACTIONS WITH ATOLL PREDICTIONS
A
1
ANY ATOLL PREDICTIONS
Y
N
SET ERROR FLAG AND INCREMENT ERROR COUNT
A
1
ANY DNS DIFFERENT ?
Y
N
COUNT AND STORE DIFFERENCES
SET ERROR FLAG AND INCREMENT ERROR COUNT
ANY ATOLL DIFF. ?
Y
N
COUNT AND STORE DIFFERENCES
SET ERROR FLAG AND INCREMENT ERROR COUNT
CHEKZ RETURN

Figure A-10. SUBROUTINE COMPAZ (1 of 1)

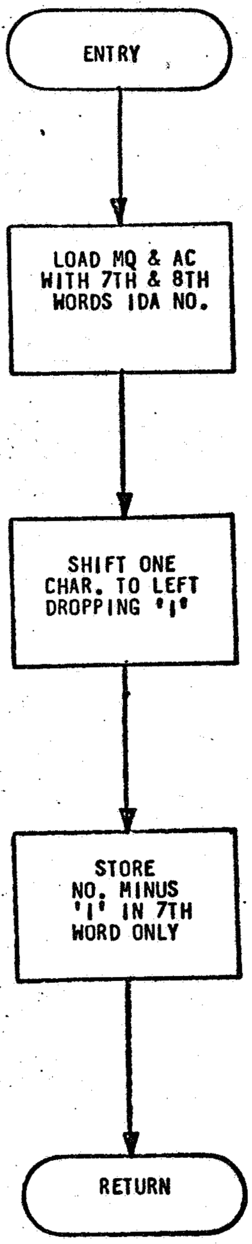

ENTRY
LOAD MQ & AC WITH 7TH & 8TH WORDS IDA NO.
SHIFT ONE CHAR. TO LEFT DROPPING 'I'
STORE NO. MINUS 'I' IN 7TH WORD ONLY
RETURN

Figure A-11. SUBROUTINE NIDEX (1 of 1)

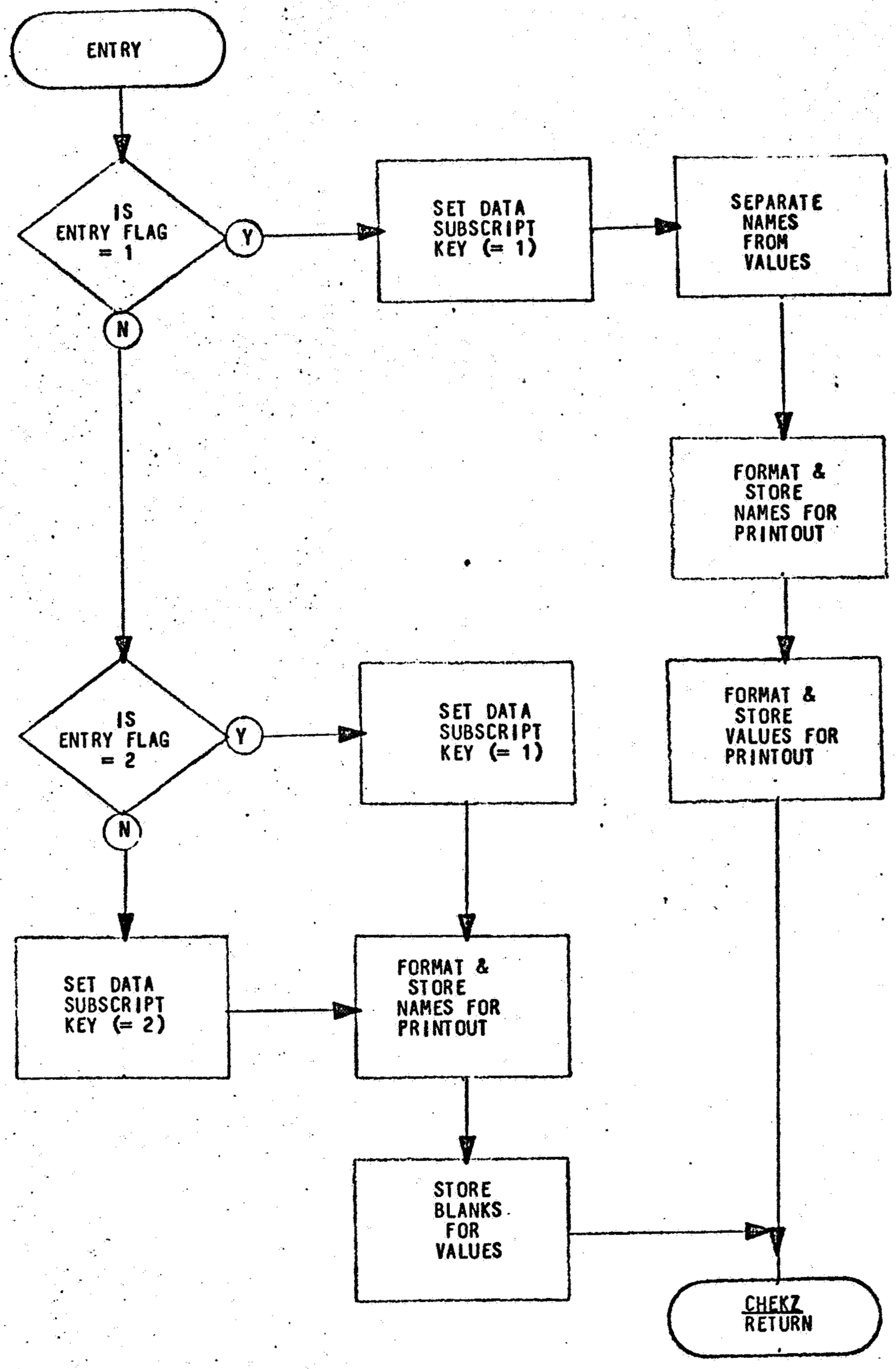

ENTRY
IS ENTRY FLAG = 1
Y
N
SET DATA SUBSCRIPT KEY (= 1)
SEPARATE NAMES FROM VALUES
FORMAT & STORE NAMES FOR PRINTOUT
FORMAT & STORE VALUES FOR PRINTOUT
IS ENTRY FLAG = 2
Y
N
SET DATA SUBSCRIPT KEY (= 1)
SET DATA SUBSCRIPT KEY (= 2)
FORMAT & STORE NAMES FOR PRINTOUT
STORE BLANKS FOR VALUES
CHEKZ RETURN

Figure A-12. SUBROUTINE PREPAZ (1 of 1)

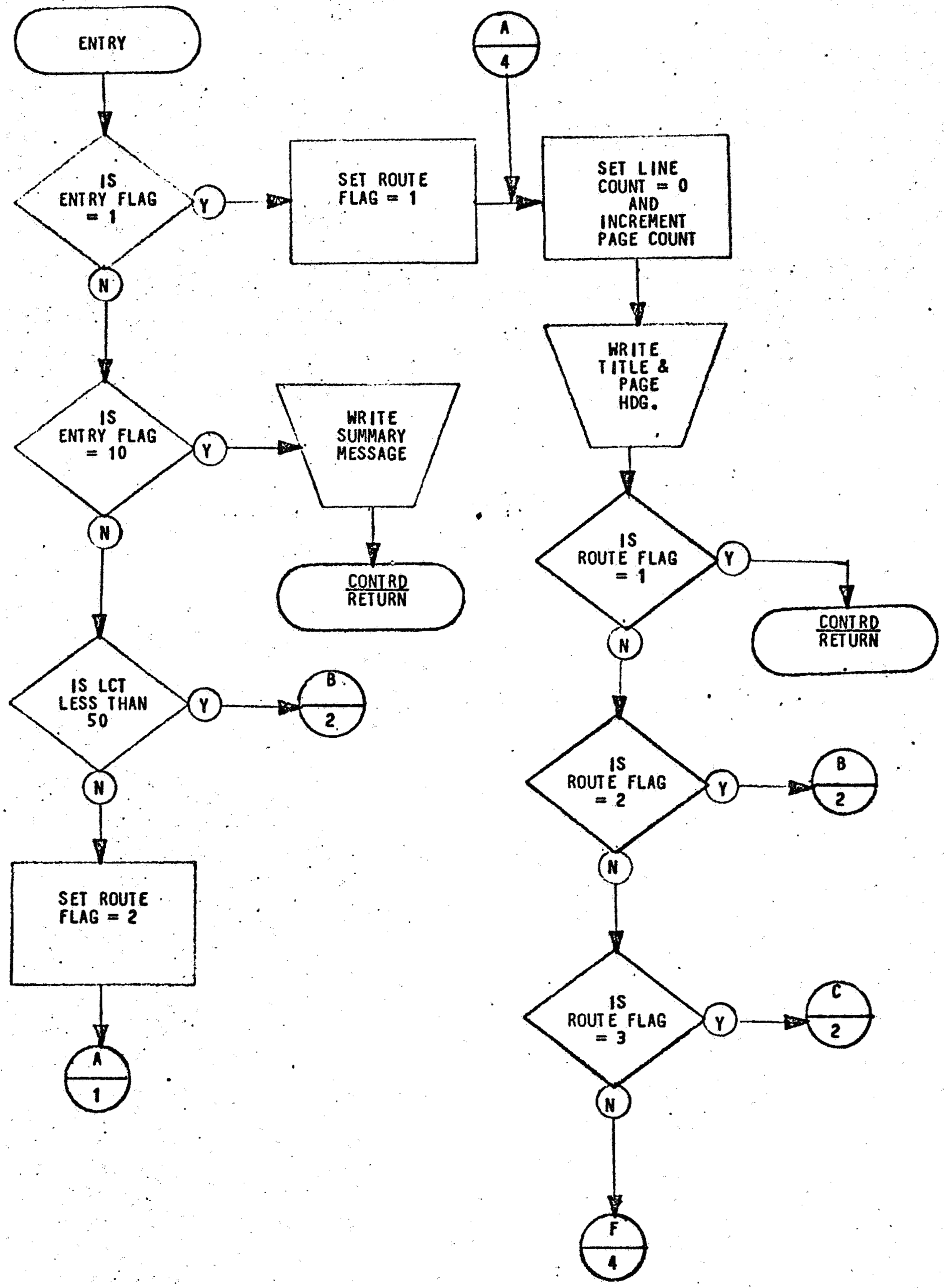


Figure A-13. SUBROUTINE PRINZ (1 of 4)

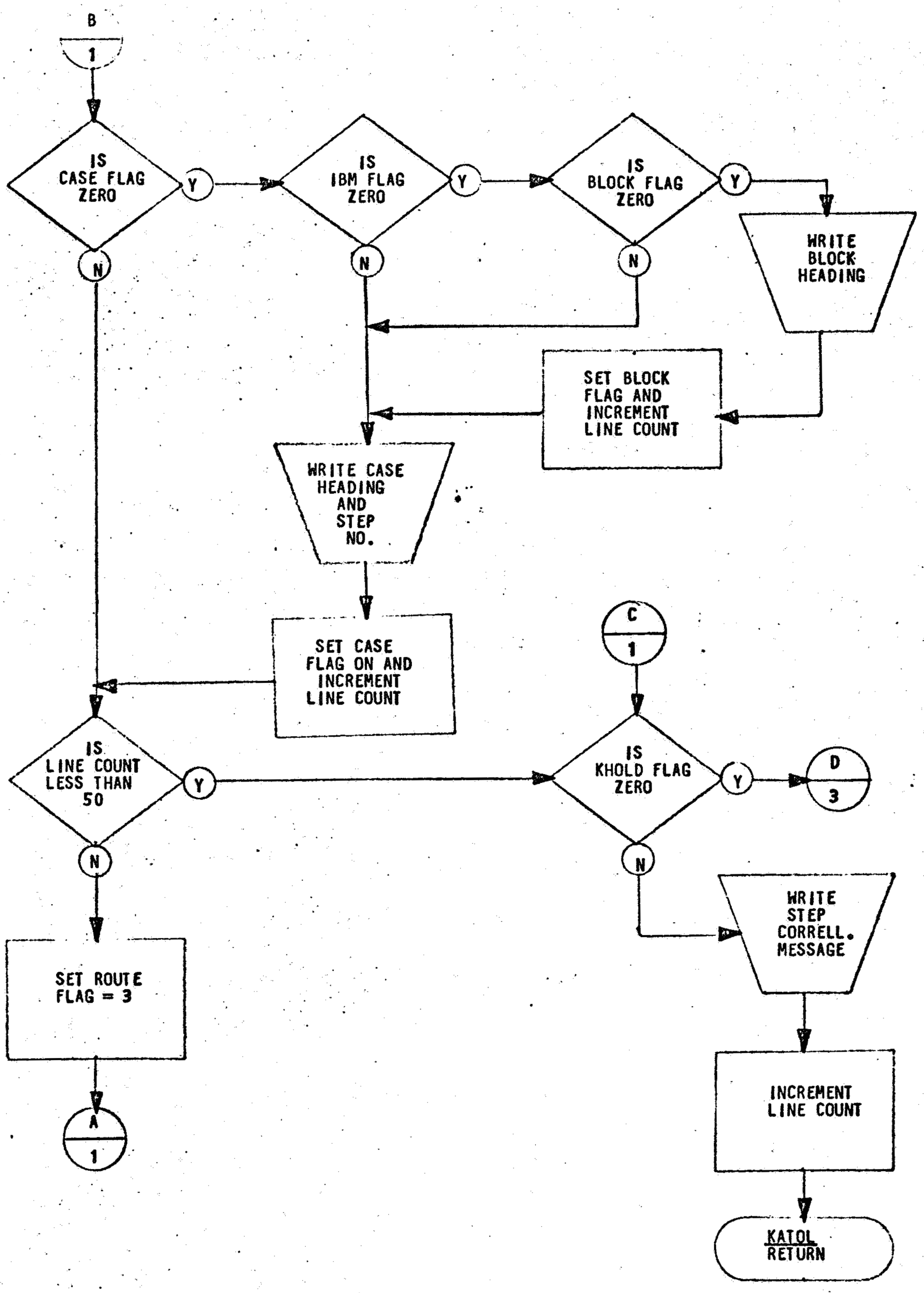

B
1
IS CASE FLAG ZERO
Y
N
IS IBM FLAG ZERO
Y
N
IS BLOCK FLAG ZERO
Y
N
WRITE BLOCK HEADING
SET BLOCK FLAG AND INCREMENT LINE COUNT
WRITE CASE HEADING AND STEP NO.
SET CASE FLAG ON AND INCREMENT LINE COUNT
C
1
IS LINE COUNT LESS THAN 50
Y
N
IS KHOLD FLAG ZERO
Y
D
3
N
SET ROUTE FLAG = 3
A
1
WRITE STEP CORRELL. MESSAGE
INCREMENT LINE COUNT
KATOL RETURN

Figure A-13. SUBROUTINE PRINZ (2 of 4)

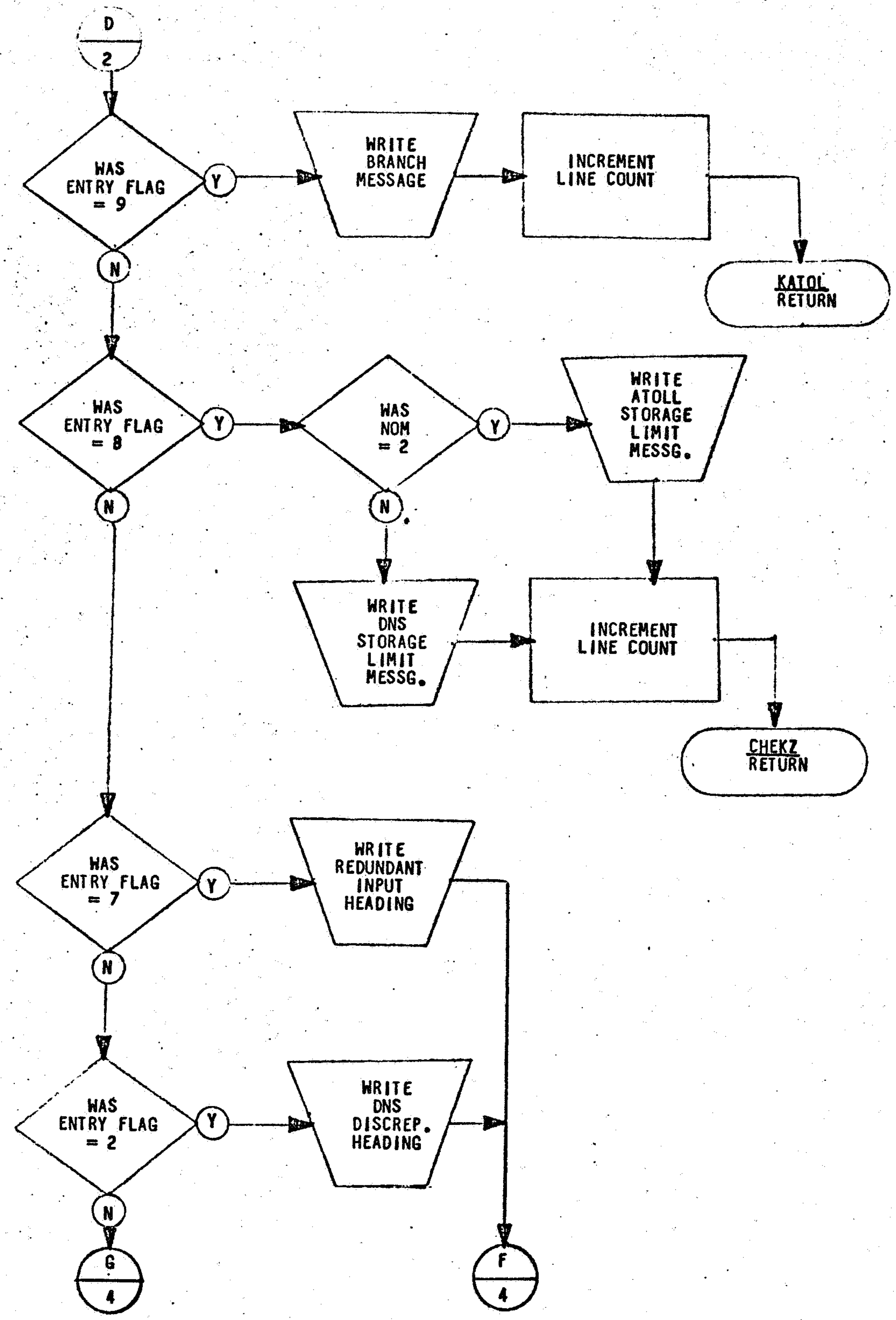


Figure A-13. SUBROUTINE PRINZ (3 of 4)

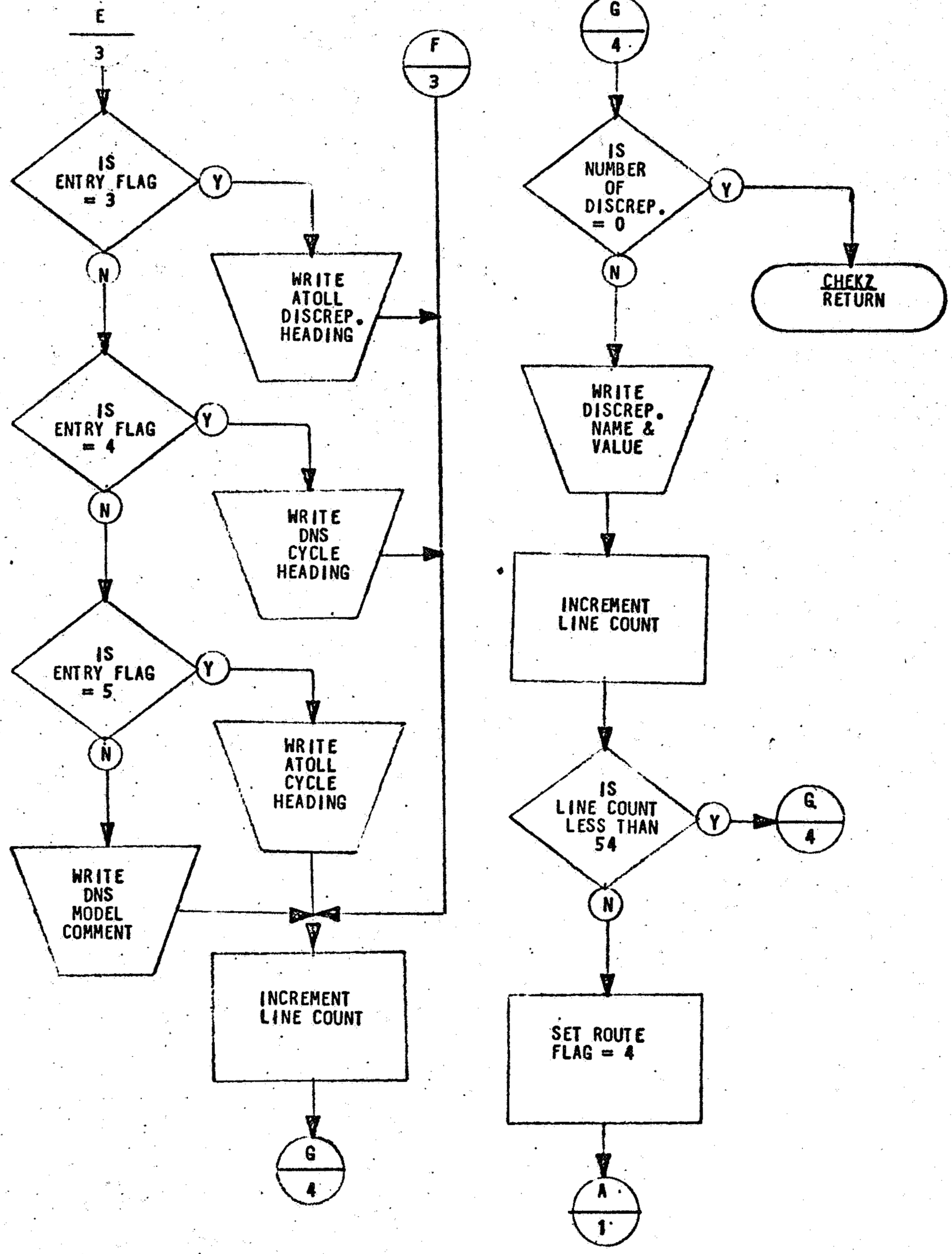


Figure A-13. SUBROUTINE PRINZ (4 of 4)

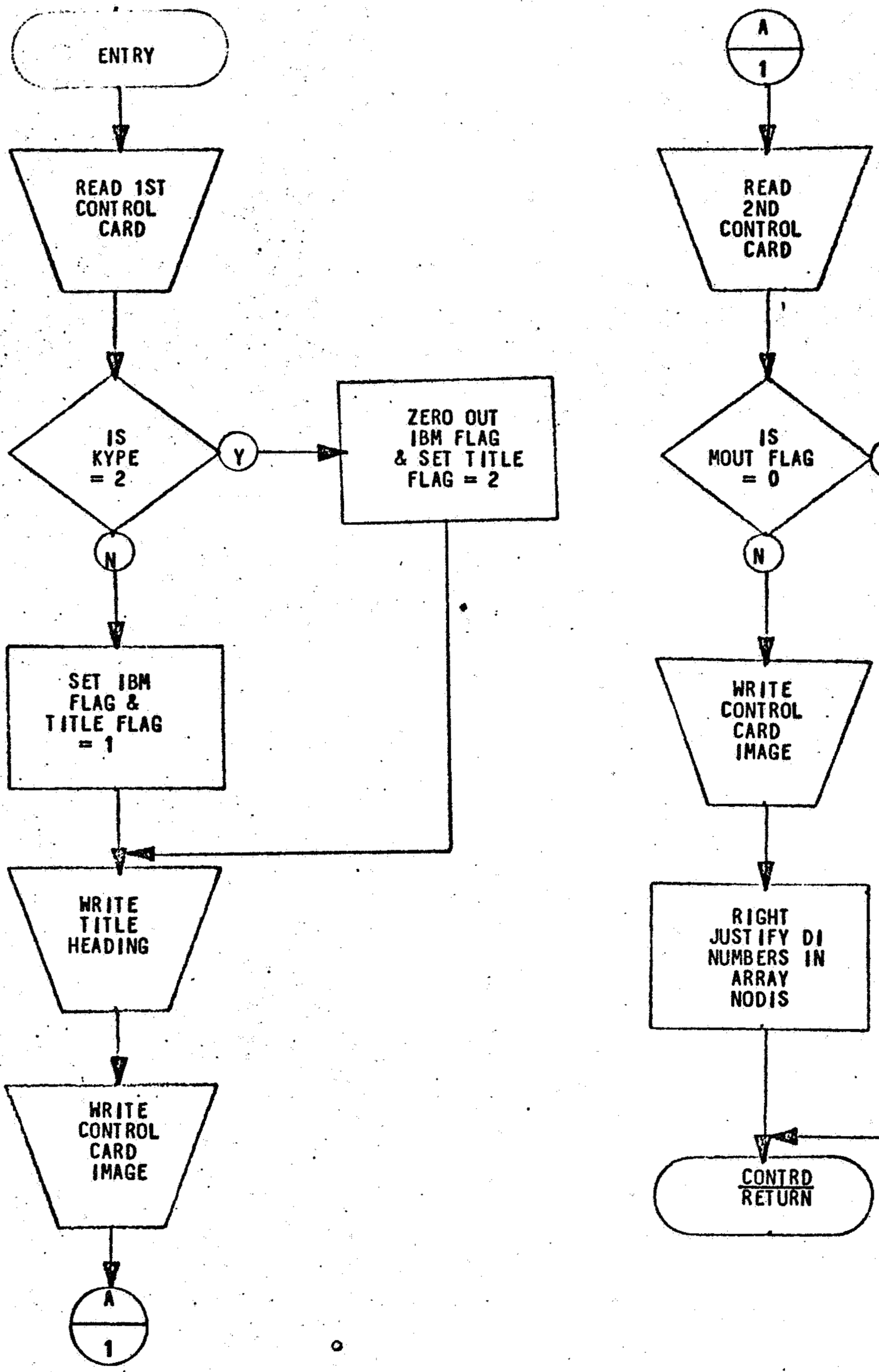


Figure A-14. SUBROUTINE READCD (1 of 1)

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLE

The following is an alphabetical listing of the terms used in the DNS/ATOLL Compator Program.

NAME	DESCRIPTION
IBMF	ATOLL Format Flag
IDA	Test Procedure ID Number
IND	Internal Directory Key
INUM	Transfer Argument
IPAG	Page Counter
ISPEC	Common Heading (JOPT, KYPE, LIST, MNT, IDA)
ISSTP	Current ATOLL Substep
ISTEP	Current ATOLL Step
JOPT	(Unused)
JSTPS	Common Heading (ISTEP, MSSTP)
KADI	ATOLL DI and Value
KADIS	Common Heading (KDIS, KADI, KLIM)
KAFG	ATOLL IDA in Progress Flag
KATRL	Common (KRDF, KONF, KENDF, KDIF, KNUM, KNUF)
KAWDS	ATOLL Tape Input Buffer
KBOT	ATOLL "BOTB" Operator
KCAS	ATOLL Case Counter
KDIF	ATOLL DI Value Flag
KDIS	ATOLL DI Counter
KEND	ATOLL "END" Operator
KENDF	ATOLL END Data Flag
KENF	ATOLL "DISI" Operator Flag
KEPCK	Current ATOLL Step/Substep
KER	ATOLL Discrepancy Counter
KERC	DNS/ATOLL Discrepancy Counter

NAME	DESCRIPTION
KHOLD	Step Correlation Flag
KINF	ATOLL "DISO" Operator Flag
KLIM	ATOLL DI Storage Limit Flag
KNAM	ATOLL "NAME" Operator
KNT	IDA'S yet to be Processed Counter
KNTRL	Common Block (KTRL, IBMF, NSFG, KAFG, LSFG)
KNUF	ATOLL "DI in Process" Flag
KNUM	Current DI Name
KONF	ATOLL Continuation Flag
KOVER	ATOLL "DI Overlook" Flag
KRDF	ATOLL "Case in Progress" Flag
KSTF	ATOLL "Step Located" Flag
KTIM	Step Discrepancy Counter
KTRL	Processing Control Key
KYC	ATOLL "Recurring DI" Counter
KYCLE	ATOLL "Recurring DI" Name
KYCLES	Common Block (KYC, KYCLE)
KYPE	IBM/Boeing Format Key
LBKF	Previous Block Number
LBLF	ATOLL Block Flag
LBRA	BRANCH Transfer Argument
LCT	Line Counter
LETF	ATOLL Letter Flag
LIST	DNS State List Flag
LOST	DI to be ignored
LPRV	Common Heading (LSDV)
LRET	Internal Return Key
LSDV	DI BCD Value
LSTEP	Case Starting Step Number
LSSTP	Case Starting Substep Number
MCAS	Common Heading (NCAS, KCAS, KERC, NTES)
MNT	Number of IDA's
MOUT	Number of DI's to be ignored
MSSTP	Branching Substep Number
NAM	Current Test Procedure
NAMS	Common Heading (NID, NAM, NBLOC, NXTB, LBLF, NSUBT)
NBLOCK	Current Block Number
NBRA	Current Branching Step/Substep
NCAS	Simulation Case Counter
NCT	List Processing Line Counter

NAME	DESCRIPTION
NDIF	Simulation DI Value Flag
NDIS	Simulation DI Counter
NDOF	Redundant Input Flag
NDOS	Redundant Input Counter
NENFG	Simulation Enter Flag
NEXT	Next Case Step/Substep Number
NGOOD	Common Heading (MOUT, NODES)
NID	Current Test Procedure ID
NINFG	Simulation Input Flag
NLIM	Simulation DI Storage Limit Flag
NNFG	Simulation Name in Process Flag
NNUM	Current Simulation Name
NOD	DI's to be ignored Counter
NODIS	Don't Care DI List
NOK	Simulation DI Discrepancy Counter
NOPRT	Common Heading (NOD, LOST, KOVER)
NPAG	List Processing Page Counter
NRDF	Simulation Case in Progress Flag
NSDI	Simulation DI and Value
NSDIS	Common Heading (NDIS, NSDI, NLIM)
NSDO	Redundant DO Name
NSDOS	Common Heading (NDOS, NSDO)
NSFG	Simulated IDA in Progress Flag
NSTEP	Current Case Step/Substep
NSTEPS	Common Heading (NSTEP, NEXT, LSTEP, LSSTP, KEPCK, KHOLD)
NSTPF	Step Label Flag
NSUBT	Case Heading Flag
NSWDS	Simulation Tape Input Buffer
NTES	DNS/ATOLL Discrepancy Flag
NUMS	Common Heading (NNFG, NNUM, NRDF, NDOF)
NXTB	Next Block Number
NXTF	Next Case Ready Flag
NYC	Simulation Recurring DI Counter
NYCLE	Simulation Recurring DI Name
NYCLES	Common Heading (NYC, NYCLE)

2.0 DEFINITIONS

NAME	DESCRIPTION
IBMF	A word in common block/ISPEC/ used by the program as a means of selecting alternate groups of instructions

NAME	DESCRIPTION
	for processing certain ATOLL format or style differences between Boeing and IBM test procedures. The flag is set (by means of control card 1) to 1 if the ATOLL tape to be used is an IBM tape. If not, the flag is zeroed.
IDA	A 10 word array in common block/ISPEC/ used to store the names of any (up to 10) test procedures which are to be processed. The array is used in conjunction with counter MNT. Each name (ID) is stored in sequence as a 6 character BCD name exactly as it appeared in the data field of control card 1.
IND	An internal keyword used in subroutine PRINZ to direct the program to specific groups of instructions. Its value is determined from the calling routine.
INUM	An internal location used in subroutine DNSINP as an argument when making a call.
IPAG	An internal location in subroutine PRINZ used for counting and numbering pages for the program printout.
ISPEC	A common block of 14 words used to store the contents of the first control card. The array contains JOPT, KYPE, LIST, MNT, AND 10 IDA locations.
ISSTP	An internal word in subroutine KEPCKZ used to store the latest test procedure substep number encountered when reading the ATOLL tape. The substep is stored as 2 BCD characters right adjusted and zero filled leading.
ISTEP	A word in common block/JSTPS/ used to store the latest test procedure step number encountered when reading the ATOLL tape. The step number is stored as a 4 character (right ADJ nonblank BCD) number left adjusted in the word and zero filled following.
JOPT	Unused spare word in common block/ISPEC/.

NAME	DESCRIPTION
JSTPS	A common block containing ISTEP and MSSTP, and accessible to PRINZ for printing information concerning branching test instructions encountered on the ATOLL tape.
KADI	A 100 word array in common block/KADIS/ used for storing the name and value of each discrete prediction encountered when processing ATOLL "DISIO" or "DISI 1" records during a case. The value is stored as two BCD characters left adjusted in the word. The numerical DI designation is stored as a 4 BCD character number right adjusted in the word. For printout, the value characters are replaced by the BCD designator "DI". The array is zeroed at the start of each case.
KADIS	A common block containing KDIS, 100 KADI locations, and KLIM. It is used for storing processed ATOLL prediction data during a case.
KAFG	A word in common block/KNTRL/ used as a flag to signal that a particular test procedure is in process on the ATOLL tape. It is set when the particular test procedure name (NID) has been located on the ATOLL tape, and zeroed out when a subsequent ATOLL "END" record is encountered.
KATRL	A common block containing 6 words or flags used for controlling the reading and processing of the ATOLL tape. It contains KRDF, KONF, KENDF, KDIF, KNUM, and KNUF.
KAWDS	A common block containing 14 words used as the input buffer for storing each 14 word ATOLL tape record as it is being processed.
KBOT	A test word in subroutine KATOL used for identifying ATOLL "BOTB" records.
KCAS	A word in common block/MCAS/ used as a counter for storing the current number of ATOLL cases which have been processed. It is zeroed at the start of each test procedure.

NAME	DESCRIPTION
KDIF	A word in common block/KATRL/ used as a flag to signal the value of the current DI predictions associated with an encountered ATOLL "DISIO" or "DISI 1" record. The flag is set minus for "DISIO" records, and is set plus for "DISI 1" records.
KDIS	A word in common block/KADIS/ used as a counter for storing the number of ATOLL DI predictions encountered during a case. It is zeroed at the start of each case.
KEND	A test word in subroutine KATLT used for identifying ATOLL "END" records.
KENDF	A word in common block/KATRL/ used as a flag to signal that all names associated with a particular ATOLL "DISIO" or "DISI 1" record have been processed. It is set whenever an ATOLL "End of variable field" is encountered or signaled.
KENF	A flag in subroutine KATLT used to indicate that an ATOLL "DISIO" or "DISI 1" record has been encountered. It is zeroed at the end of each case.
KEPCK	A word in common block/NSTEPS/ used as a means of identifying specific ATOLL records which are to be treated as the starting and ending records for a case. The end word is updated in subroutine KEPCKZ each time a new ATOLL step or substep is encountered. It is formed by combining the current or latest step (ISTEP) and the current substep (ISSTP), with the step number occupying the leftmost 4 BCD characters, and the BCD substep number occupying the remaining 2 characters.
KER	An internal counter in subroutine COMPAZ used as a counter and subscript for storing any ATOLL predictions which were not found in the simulation results for a case.

NAME	DESCRIPTION
KERC	A word in common block/MCAS/ used as a counter for storing the total number of discrepancies encountered during a complete DNS/ATOLL test procedure comparison. (If no discrepancies occurred, a comment so stating will be printed out at the end of that comparison.
KHOLD	A word in common block/NSTEPS/ used as a flag to signal that an ATOLL step/substep number higher than the number specified as a case ending record has occurred. If a step/substep discrepancy occurs, the flag is set to 1, and the current case is concluded, but the comparison is bypassed. If a match between the ATOLL "KEPCK" and SIMULATION "NEXT" occurs during the following case, the flag is set to 2, and correlation is re-established. (If no match is located after 10 attempts, the program dumps and exits).
KINF	An internal flag in subroutine KATLT. It is set to 1 if an ATOLL "DISO 0" or 'DISO 1" record is encountered. It is zeroed at the start of each case, and was included for future program capability expansion.
KLIM	A word in common block/KADIS/ used as a signal flag should the specified storage limit of 100 ATOLL predictions for one case be exceeded. If exceeded, each excess DI encountered is stored in the flag.
KNAM	A test word in subroutine KATLT used to identify ATOLL "NAME" records.
KNT	An internal counter and test word in driver program CONTRD used to keep track of the number of test procedures remaining to be processed. It is set with the number from control card 1 (MNT), and tested before each test procedure is started. (If zero, the program exits. If not, KNT is decremented and a test procedure is set up for comparison).

NAME	DESCRIPTION
KNTRL	A common block containing 5 process control flags (KTRL, IBMF, NSFG, KAFG, LSFG).
KNUF	A word in common block/KATRL/ used as a flag and temporary character storage cell during the identification of names encountered in the ATOLL variables field. It is set in KATRG when any of the characters (0 thru 9) are detected, and zeroed whenever any other character is encountered.
KNUM	A word in common block/KATRL/ used as temporary storage while building up the numerical name designation for discrete predictions encountered while processing the variables field of ATOLL "DISIO" or "DISI 1" records. The word is formatted in BCD, and will contain a 4 character numerical discrete designation right adjusted and zero filled.
KONF	A word in common block/KATRL/ used as a flag to signal that the current ATOLL variables field is to be continued on the next ATOLL record. It is set in subroutine KATRG if a continuation character is encountered, or no field termination characters are encountered. It is zeroed before each new variables field is processed.
KOVER	A word in common block/NOPRT/ which is set in KATRG during initial entry to signal all subsequent entries during the run that any required initialization of data pertaining to DI's to be ignored has been completed.
KRDF	A word in common block/KATRL/ which is set in subroutine KATRG during the first entry in a case. It signals any subsequent entries that initialization of controls for DI predictions data has been completed. It is re-zeroed at the end of each case.
KSTF	An internal word in subroutine KATLT used as a flag to signal that the case starting step has been located on the ATOLL tape and the case is ready to process. When the correct starting step (NSTEP)

NAME	DESCRIPTION
	is located, the flag is set to 1. It is maintained at 1 each time the correct case ending step (NEXT) is located. It is zeroed at the beginning of each new test procedure.
KTIM	An internal word in subroutine KATLT used as a counter to store the number of times a step correlation discrepancy has occurred (KHOLD = 1). (If the count exceeds 10, the program dumps and exits).
KTRL	A word in common block/KNTRL/ used as a processing status flag. Its value is returned to CONTRD as 2 for all normal returns. If its value is 10, a processing error has been detected, and the program is set to dump and exit.
KYC	A word in common block/KYCLES/ used as a counter to store the number of ATOLL discretes which were encountered more than once during any one case. The counter does not increment above 10, and is zeroed at the beginning of each case.
KYCLE	A 10 word array in common block/KYCLES/ used to store the names of any ATOLL discretes which were encountered more than once during a case. If more than 10 recurring names are encountered, the last name encountered will be placed in the 10th word. The array is zeroed at the start of each case.
KYCLES	A common block containing a counter KYC and 10 KYCLE locations used for storing any recurring ATOLL DI predictions which are encountered during a case.
KYPE	A word in common block/ISPEC/ used to signal the value to be assigned to ATOLL format flag IBMF. The value assigned to KYPE is taken from control card 1.
LBKF	An internal word in subroutine KATLT used with Boeing ATOLL tape to signal that the ATOLL block record corresponding to simulation block "NBLOC" has occurred. It is zeroed at the start of each test procedure.

NAME	DESCRIPTION
LBLF	A word in common block/NAMS/ used as a flag in subroutine PRINZ to signal whether the current test block heading has been printed out when an ATOLL tape using the Boeing format is being used.
LBRA	A word in subroutine KATLT used as an argument when calling subroutine PRINZ with information concerning branching ATOLL test instructions.
LCT	A word in subroutine PRINZ used as a line counter for controlling the paging and printing of discrepancy lists. It is zeroed whenever 54 lines have been printed.
LETF	A word in subroutine KATRG used to store and signal when any character other than a number or a control character has been encountered while processing the variables field of an ATOLL record. It is zeroed whenever a number or a control character is encountered.
LIST	A word in common block/KNTRL/ used to signal CONTRD and DNSINP that the simulation state and cycle list contained on the simulation tape is to be processed at the end of the current test procedure comparison. Its value is set to 1 at run time from control card 1 if the state list is to be processed. If not, it is set to 0. Its value holds for the entire run.
LOST	A 40 word array in common block/NOPRT/ used to store any ATOLL predictions encountered which were included at run time in a pre-specified list of DI's to be ignored. The array and its associated counter, NOD, are zeroed at the beginning of each case.
LPRV	A common block containing 100 LSDV locations used for storing the BCD value data associated with ATOLL or simulation discretes when printing discrepancy lists.

NAME	DESCRIPTION
LRET	An internal word in subroutine PRINZ which is set to selected values if line counter LCT exceeds 54 when tested at various processing points. The setting provides a means for controlling a transfer to the paging and title section and a return to the processing point.
LSDV	An array of 100 words in common block/LPRV/ which is used for storing the BCD value data associated with ATOLL or simulation discretes when printing discrepancy lists. The BCD words may contain blanks, or the statement "=1.", or the statement "=0.".
LSSTP	A word in common block/NSTEPS/ used to store the case starting substep formatted in BCD for print-out if a case heading is required. It is processed in subroutine CONVRT from the current contents of NSTEP.
LSTEP	A word in common block/NSTEPS/ used to store the case starting step formatted in BCD for print-out when a case heading is required. It is processed in subroutine CONVRT from the current contents of NSTEP.
MCAS	A common block containing 4 storage locations for case data. (NCAS, KCAS, KERC, NTES).
MNT	A word in common block ISPEC/ used for storing the number of test procedures which are to be processed during the run. Its value is set in subroutine READCD when control card 1 is read.
MOUT	A word in common block/NGOOD/ used for storing the number of DI's which are to be ignored during the run. Its value is set in READCD when control card 2 is read, and can be from 0 to 200.
MSSTP	A word in common block/JSTPS/ used for storing the current substep number formatted in BCD for print-out if information concerning branching ATOLL instructions is required. Its value is set in subroutine KEPCKZ from the current contents of KAWDS.

NAME	DESCRIPTION
NAM	A word in common block/NAMS/ used for storing the name of the current test procedure. It is set in subroutine DNSINP, and is formatted in BCD for use when printing out the title heading.
NAMS	A common block containing 6 storage locations for test procedure identification and status data. (NID, NAM, NBLOC, NXTB, LBLF, NSUBT)
NBLOC	A word in common block/NAMS/ used for storing the starting test block number. Its value is set in subroutine DNSINP with data from the simulation "BLOCK" record, and is formatted in BCD for use when printing out block headings in subroutine PRINZ.
NBRA	An internal word in subroutine KATLT used to process ATOLL test records for branch data, and as an argument when calling subroutine PRINZ.
NCAS	A word in common block/MCAS/ used as a counter for storing the current number of simulation cases which have been processed. It is zeroed at the start of each test procedure.
NCT	An internal word in subroutine DNSINP used as a line counter, and used for controlling the paging and titling of the printout when a simulation state line is processed.
NDIF	An internal word in subroutine DNSRDG used as a flag to signal the value of a simulation discrete encountered on the current record. The flag is set negative if the current DI value is 0.
NDIS	A word in common block/NSDIS/ used as a counter for storing the number of simulation DI's encountered during a case. It is zeroed at the start of each case.

NAME	DESCRIPTION
NDOF	A word in common block/NUMS/ which is set in subroutine DOSRDG during the first entry in a case to signal any subsequent entries that initialization of controls for redundant input data has been completed. It is rezeroed at the end of each case.
NDOS	A word in common block/NSDOS/ used as a counter for storing the number of DO's. during a case which were encountered on simulation records labelled "extra". It is zeroed at the start of each case.
NENFG	An internal flag in subroutine DNSINP used to signal that a simulation record labelled "ENTER" has been encountered. It is zeroed at the end of each case.
NEXT	A word in common block/NSTEPS/ used as the basic control word for correlating the starting and ending of the SIMULATION and ATOLL cases. Its value is set in subroutine DNSINP from data in word 5 of the first SIMULATION "STEP" record encountered after a case has been started. The contents of NEXT are then used by subroutine KATLT to compare with KEPCK in order to locate the equivalent case ending record on the ATOLL tape. The contents of NEXT represent the end of the current case, and the starting point for the next case. When comparison of the current case is completed, the contents of NEXT are used to update NSTEP for starting the new case. NEXT is formatted in BCD and contains a 4 character step number left ADJ and a 2 character substep number right adjusted.
NGOOD	A common block containing a count MOUT and a 200 word array NODIS. It is used if a list of DI's to be excluded from comparison is submitted with the run. Data is obtained from control card 2 (and continuations if required).
NID	A word in common block/NAMS/ used to store the identification word for the current test procedure in process. Data for the word is obtained from the current position in array IDA, and is formatted in BCD.

NAME	DESCRIPTION
NINFG	An internal flag in subroutine DNSINP used for signalling that a SIMULATION record labelled "INPUT" or "EXTRA" has occurred. It is zeroed at the end of each case.
NLIM	A word in common block/NSDIS/ used as a signal flag if the specified storage limit for 100 DI's in a case is exceeded. If exceeded, each excess DI encountered is stored in the flag.
NNFG	A word in common block/NUMS/ used as a flag and temporary character storage cell during the processing of discrete names on SIMULATION records. It is set in DNSRDG or DOSRDG when a BCD character 9 or less is detected, and zeroed when any character 10 or greater is detected.
NNUM	A word in common block/NUMS/ used as temporary storage while building up the numerical name designations for discretes encountered on SIMULATION records. It is used in DNSRDG for processing DI's, and in DOSRDG for processing redundant DO's. The word is formatted in BCD, and will contain a 4 character numerical discrete designation right adjusted and zero filled.
NOD	A word in common block/NOPRT/ used as a counter for storing the number of discretes encountered which at run time were included in a prespecified list of DI's to be ignored. The word is zeroed at the start of a case, and is used in conjunction with array LOST.
NODIS	A 200 word array in common BLOCK/NGOOD/ used in conjunction with MOUT for storing the numerical designations of any discretes which are to be excluded from comparison. The data is processed in subroutine READCD from data on control card 2 (and continuations if required). Each word used in the array will be formatted in BCD and will contain a 4 character numerical discrete designation right adjusted and zero filled. The array is unchanged during a run.

NAME	DESCRIPTION
NOK	An internal word in subroutine COMPAZ used as a counter for storing the number of simulation discretes which were not corroborated by ATOLL predictions. It is used to reset the counts in NDIS or NYC after comparison is completed.
NOPRT	A common block of 42 words reserved for counting and storing any ATOLL predictions which were included at run time in a prespecified list of DI's to be ignored. It contains counter NOD, a 40 word array LOST, and an entry flag KOVER.
NPAG	A word in subroutine DNSINP used as a page counter for use when a simulation state list is processed for printout.
NRDF	A word in common block/NUMS/ which is set in subroutine DNSRDG during the first entry in a case. It signals subsequent entries that initialization of controls for storage of simulation DI data has been completed. It is rezeroed at the end of each case.
NSDI	A 100 word array in common block/NSDIS/ used for storing the name and value of each DI encountered when processing simulation "ENTER" records during a case. Each DI is assigned a word in sequence. The DI value is stored as 2 BCD characters left adjusted in the word. The numerical DI designation is stored as 4 BCD characters right adjusted in the word. For printout, the value characters are replaced by the designator "DI". The array is zeroed at the start of each case.
NSDIS	A common block containing NDIS, 100 NSDI locations, and NLIM. It is reserved for storing processed SIMULATION discrete data during a case. It is the SIMULATION counterpart of the ATOLL KADIS block.

NAME	DESCRIPTION
NSDO	A 20 word array in common block/NSDOS/ which is reserved for storing the numerical designations of DO's encountered on SIMULATION "EXTRA" records during a case. It is used in conjunction with counter NDOS, and is zeroed at the start of each case.
NSDOS	A common block containing counter NDOS and 20 word array NSDO. It is reserved for counting and storing discrete output data encountered on simulation records labelled "extra" during a case.
NSFG	A word in common block/KNTRL/ used as a flag to signal that a particular test procedure is in process on the simulation tape. It is set when the particular test procedure name (NID) has been located on the simulation tape, and is zeroed out when a subsequent record labelled "END" is encountered.
NSTEP	A word in common block/NSTEPS/ used as the initial case starting correlation word when processing a test procedure. Its initial value is obtained from the first SIMULATION "STEP" record encountered and is subsequently used in subroutine KATLT to locate the equivalent initial case starting record on the ATOLL tape. The value is updated with the current contents of location NEXT when each subsequent SIMULATION case is started. The word is formatted in BCD and contains a 4 character step number left adjusted and a 2 character substep number right adjusted.
NSTEPS	A 6 word common block containing NSTEP, NEXT, LSTEP, LSSTP, KEPCK, KHOLD. It is used for storing test procedure step and substep data, and for correlating the processing of a case from the SIMULATION tape and from the ATOLL tape.
NSTPF	An internal flag in subroutine DNSINP used to signal that the initial case is started. It is set when the first SIMULATION "STEP" record is encountered at the start of the test procedure. It is zeroed at the start of each test procedure.

NAME	DESCRIPTION
NSUBT	A word in common block/NAMS/ which is used as a flag to signal if a subtitle for the current case is required. It is set to zero at the start of a case. The flag is set to 1 in subroutine PRINZ during the first entry in a case. It signals subsequent entries in the case that the case heading has been printed.
NSWDS	A common block of 20 words used as the input buffer for storing each 20 word SIMULATION tape record as it is being processed.
NTES	A word in common block/MCAS/ used as a flag to signal if a discrepancy has been detected. It is set with the value 1 in subroutine COMPAZ if a discrepancy is uncovered. The flag is then tested in subroutine CHEKZ to determine if a discrepancy listing will be required. It is rezeroed after each discrepancy list is processed.
NUMS	A common block of 4 words used for temporary storage while processing discrete names encountered in SIMULATION records. It consists of NNFG, NNUM, NRDF, and NDOF.
NXTB	A word in common block/NAMS/ used as temporary storage for ATOLL block number data when processing SIMULATION "BLOCK" records for Boeing test procedures. The word is formatted in BCD.
NXTF	An internal word in subroutine KATLT used as a flag to signal a case entry that the ATOLL case starting record was already read in and stored in KAWDS at the end of the preceding case.
NYC	A word in common block/NYCLES/ used as a counter to store the number of SIMULATION discretes which were encountered more than once during a case. The counter does not increment above 10, and is zeroed at the start of each case.

NAME	DESCRIPTION
NYCLE	A 10 word array in common block/NYCLES/ used to store the names of any SIMULATION discretes which were encountered more than once during a case. If more than 10 recurring names are encountered, the last name encountered will be placed in the 10th word. Each word stored is in BCD format. The array is zeroed at the start of each case.
NYCLES	A common block containing a counter NYC and a 10 word array NYCLE. It is used for counting and storing any simulation DI's which appear more than once during a case.

SECTION

5

DNS TRANSLATOR PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

5

DNS TRANSLATOR PROGRAM

AUTHOR:	A. R. Stone Convair division of General Dynamics
PURPOSE:	The DNS Translator Program substitutes expanded definitions or actual hardware nomenclature for the coded variable names contained on the output tape from a previous test procedure simulation. It produces a translated printout and, if desired, a translated copy of the output tape.
RESTRICTIONS:	1. The programs must run on an IBM 7094 with IBJOB systems capability. 2. In addition to system input and output, three magnetic tape units are required. (Two for BCD input use, and one for BCD output use). 3. The maximum number of dictionary names which can be supplied is 4000. The maximum size for any name definition is 24 BCD characters.
STORAGE:	The program and its associated buffer storage area extends consecutively from core location $3064)_8$ to $67722)_8$. The program consists of the following three sub-programs:

1.	DNSINP	Driver, and simulation tape reading and writing routine (Fortran IV).
2.	DNSRDG	Locates simulation names in dictionary list (MAP).
3.	READC	Reads and stores a dictionary of names vs definitions (Fortran IV).

TIMING:	The program processes approximately 1000 to 2000 records per minute depending on control options specified.

USE:

1. A simulation output tape, a control card, and an input dictionary containing names versus desired definitions are required to set up the program for operation. The simulation output tape, previously created during a simulation run, contains a history of actions and reactions, and state lists, resulting from stimulating the DNS model with inputs derived from an appropriate test procedure. The code names which appear in the simulation output are essentially 'nicknames' used in place of the full names of system hardware (locatable on the schematics) when the DNS Model was constructed. These code names are limited to one IBM word for simplicity and conservation of computer memory. In order to make the results more meaningful to users interested in a greater level of detail than comparison of DO/DI relationships, the DNS Translator Program permits reverting from the 'nicknames' back to fully locatable hardware names of up to twenty-four characters.

2. A typical operational card deck setup is:

```
$JOB
$PAUSE
$ATTACH          A5  (Output save tape)
$AS              SYSCK1, HI
$ATTACH          B6  (simulation results)
$AS              SYSUT6, HI
$ATTACH          A7  (dictionary tape, if used)
$AS              SYSUT7, HI
$EXECUTE         IBJOB
$IBJOB           GO

       (DNS Translator Binary Program Deck)

$DATA
*NAMES

       (Dictionary Card Deck)

*END NAMES
7/8 (EOF)
```

3. Control card usage is as follows:

*INPUT	This card is only used if the dictionary card images have previously been stored on a special input tape. If used, it will be the only data card, and will be followed immediately by the 7/8 end of file card. The control label, *INPUT is in card columns 1 thru 6. The asterisk must be in column 1.
*NAMES	This card signals the start of the input dictionary, and must be present.

The card format is:

Col 1 thru 6	*NAMES
Col 7 thru 12	BLANK
Col 13 thru 18	May be blank, or may contain LIST 0 or HISTOR. LIST 0 is used if only the state lists are to be translated. HISTOR is used if only the simulation history is to be translated. If blank, both the history and lists will be processed.
Col 19 thru 24	May contain anything
Col 25 thru 30	May be blank, or may contain "B1". If columns 25 and 26 contain "B1", a flag is set to signal that the tape being translated is a standard system print which already contains titles and page numbers. If blank, the normal simulation output tape will be translated, and titling and paging control will be used for the printout.
Col 31 thru 80	Blank

The input dictionary cards immediately follow the *NAMES card, and contain one variable per card in the following format:

Col 1 to 6	The code name for the variable.
Col 7 to 12	Blank
Col 13 thru 36	As many columns as required to fully describe the variable.
Col 37 thru 80	Blank

*END NAMES	This card signals the end of the input dictionary, and must immediately follow the last dictionary card. The control label *END N must be in columns 1 thru 6.

METHOD:

1. The program first reads the cards following the *DATA card. If the first card contains the control label *INPUT, the program initializes an input tape unit and sets a flag to enable reading the dictionary cards (surrounded by the *NAMES/*END NAMES cards) from the special input tape, and proceeds to 2. If the first card does not contain the *INPUT label, the program proceeds to 3 and continues processing cards from the standard load tape.

2. A card image is read in.

3. If the names in progress flag has already been set, the program goes immediately to 4 to process a dictionary card. If the flag is not set, the program checks for the *NAMES label. If the *NAMES label is absent, the program returns to 2 for the next card. When the *NAMES label is encountered, words 3 and 5 are checked for processing labels, and appropriate control flags are set to perform the specified processing of the simulation histories, simulation state lists, and output save tape. The 'names in progress' flag is set, and the program returns to 2 to begin processing the input dictionary.

4. If the *END NAMES label is encountered, the 'names in progress' flag is reset to a final value (2), and the program proceeds to 5 to process the simulation results. If the *END NAMES label is not present, a name counter is incremented, and the code name from word 1 is stored in the next position in a NAME list. The four word definition field is checked, and if a definition has been included, the data is transferred to the four word group assigned to this code name in array NTRAN. If no definition was included, the code name itself is used as the definition, and is transferred to the assigned group in NTRAN. The program then returns to 2 for the next card.

5. A record from the simulation results tape is read in.

6. If the "LIST" flag is zero (has not been set), the program proceeds to 7. If the LIST flag has been set, the program skips to 12.

7. If the ID flag (INUM) has been set, the program proceeds to 8. If the ID flag is zero, and the record contains the *NAME label, the ID flag is set before the program proceeds to the print routines in 10.

8. The record is checked to see what simulation label it contains. If it contains a simulation history label and the history control flag has been set, the program proceeds to 9. If the history control flag is zero, the program skips to 10. If it contains the state list label, the program proceeds to 11. If it contains the "End of Simulation" label, the "ID" flag value is set to signal completion of the translation, and the program then proceeds to 10. If it contains no label, the program proceeds directly to 10.

9. The code name is processed for translation. The code name (of the variable) contained on the simulation record is compared to the list of code names in the NAME list previously processed in 3. If the code name is located, the corresponding 4 word group from NTRAN is substituted for the code name on the record, and the program proceeds to 10. If no match is found, the program proceeds directly to 10.

10. The print controls are checked. If an output save tape is to be created, the processed simulation record is written on the save tape before continuing. The input tape flag is checked, and if a standard print tape is being translated, paging and title controls are bypassed. If the tape is the simulation output tape, the line count is checked and if zero, paging and title printing is accomplished before writing the processed simulation record for the standard system print. The input tape flag is again checked, and if a standard print tape is being translated, line count incrementing and checking is bypassed. If the tape is the simulation output tape the line count is incremented, checked and zeroed if it has reached 50. The ID flag is checked and if it has been set to signal the end of the translation, (Refer to 8), the program cleans up and exits. If not, the program returns to 5.

11. The state list controls are set. If the list control flag is zero, the list is not to be translated, so the program skips directly to 10. If the list control flag has been set, the 'List in Progress' flag is set before proceeding to 10.

12. The state list is processed. If the record contains the *END LIST label, the 'list in process' flag is zeroed, and the program skips to 10. If the record is not blank, the program proceeds to 9. If blank, the program goes directly to 10.

INPUT FORMATS: Figure 5-1 represents the format of the Simulation Output Tape and standard output print from a portion of a typical simulation run. The variable names associated with the simulation history labels "input" and "enter", are representative of the "nicknames" used when constructing the DNS model. The vertical bar has been added to highlight variables whose dictionary cards have been included in Figure 5-2. These variables are highlighted to demonstrate the translation technique for obtaining the output shown on Figure 5-3.

Figure 5-2 represents part of an input dictionary which will be used in conjunction with the Translator Program to reprocess the Simulation Output Tape. The input dictionary may be in the form of a card deck, or it may be an input tape containing card images of the input deck. The first word on the card is the code name or 'nickname'. The third, fourth, fifth, and sixth words are available for the associated definition. These cards represent the results of adding definitions to the basic name cards created previously by an associated program (reference Section 2 , DNS Name/ Time card Generator). The card for variable DI122 is typical of a basic name card. Since no definition has been added in for this variable, its code name would be used as its translation name.

OUTPUT FORMAT: Figures 5-3 and 5-4 represent the format of the standard output print after translating the Simulation Output Tape of Figure 5-1. The vertical bar has been added to highlight the variables whose dictionary cards were included among the group shown in Figure 5-2. A comparison of Figures 5-1 and 5-3 shows that the only discernible changes are to the names of variables which have a definition included in the dictionary. If no dictionary card was supplied, or the definition field for the variable was left blank, the output record will be identical to the input record.

```
*HEAD LIST
INPUT     DØ179     1   8
ENTER     DI176     0  13
ENTER     DI177     0  12
ENTER     DI178     0  11
INPUT     DØ171     1   7
INPUT     DØ173     1   8
ENTER     DI171     1   9
ENTER     DI173     1   9
ENTER     DI170     1  11
ENTER     DI172     1  10
INPUT     DØ220     0   5
INPUT     DØ221     0   4
ENTER     DI124     1   9
ENTER     DI125     1   8
ENTER     DI903     1   7
ENTER     DI902     1   6
ENTER     DI372     1   5
ENTER     DI159     1   4
INPUT     DØ217     0   3
INPUT     DØ218     0   2
ENTER     DI904     1   5
ENTER     DI153     1  17
ENTER     DI154     1  16
ENTER     DI155     1  15
ENTER     DI184     1  14
ENTER     DI185     1  13
ENTER     DI186     1  12
ENTER     DI905     1  11
ENTER     DI906     1  10
ENTER     DI138     1   9
ENTER     DI139     1   8
ENTER     DI187     1   6
ENTER     DI121     1   5
ENTER     DI122     1   4
ENTER     DI123     1   3
```

Figure 5-1. SIMULATION HISTORY PRINTOUT

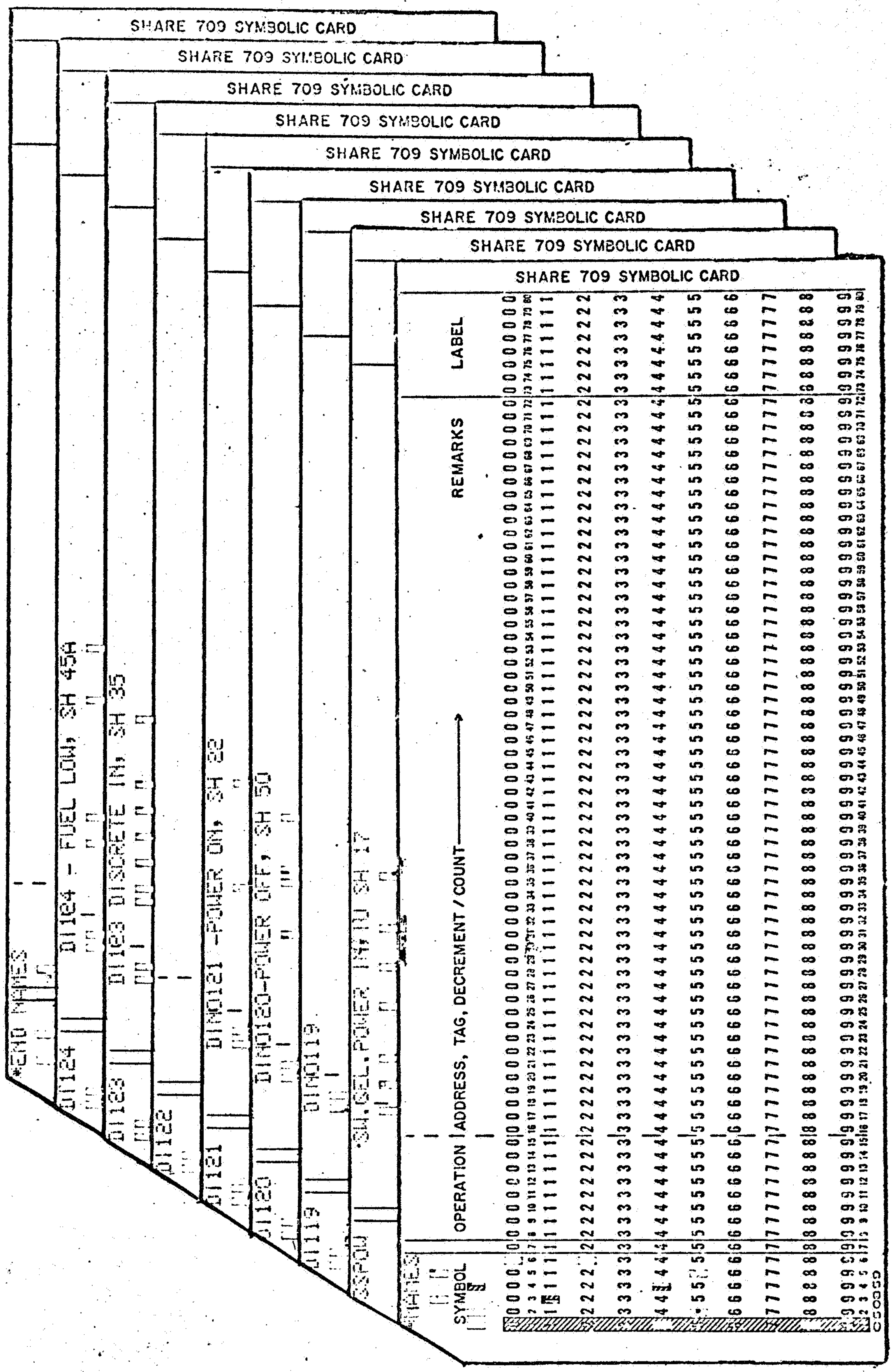

SHARE 709 SYMBOLIC CARD
SYMBOL
OPERATION
ADDRESS, TAG, DECREMENT / COUNT
REMARKS
LABEL
*END NAMES
DI124 DI124 - FUEL LOW, SH 45A
DI123 DI123 DISCRETE IN, SH 35
DI122
DI121 DINO121 -POWER ON, SH 22
DI120 DINO120-POWER OFF, SH 50
DI119 DINO119
SSPOW *SW.SEL.POWER IN, RU SH 17
NAMES

Figure 5-2. Dictionary card deck (typical).

```
*      *STEP NØ      133500
INPUT          DØNØ217                                                1     1
ENTER          DINØ138                                                0     6
ENTER          DINØ139                                                0     5
ENTER          DINØ902                                                0     4
ENTER          DINØ153                                                0    18
ENTER          DINØ154                                                0    17
ENTER          DINØ155                                                0    16
ENTER          DINØ184                                                0    15
ENTER          DINØ185                                                0    14
ENTER          DINØ186                                                0    13
ENTER          DINØ904                                                0    12
ENTER          DINØ905                                                0    11
ENTER          DINØ906                                                0    10
ENTER          DINØ159                                                0     7
ENTER          DINØ902                                                1     6
ENTER          DINØ187                                                0     5
ENTER          DINØ159                                                1     5
ENTER          DINØ121 -PØWER ØN, SH 22                               0     4
ENTER          DI122                                                  0     3
ENTER          DI123 DISCRETE IN, SH 35                               0     2
```

Figure 5-3. Translated simulation history printout.

```
*LIST  AT          8845  NØ.    2      TPID     0    0    0    0        CYCLES
    1       DINØ113                     =            0.                     0
    2       DINØ114                     =            0.                     0
    3       DINØ115                     =            0.                     0
    4       DINØ116                     =            0.                     0
    5       DINØ117                     =            0.                     0
    6       DINØ118                     =            0.                     0
    7       DINØ119                     =            0.                     0
    8       DINØ120-PØWER ØFF, SH 50    =            0.                     0
    9       DINØ121 -PØWER ØN, SH 22    =            1.                     5
   10       DI122                       =            1.                     5
   11       DI123 DISCRETE IN, SH 35    =            1.                     5
   12       DI124 - FUEL LØW, SH 45A    =            1.                     1
   13       DINØ125                     =            1.                     1
   14       DINØ126                     =            0.                     0
   15       DINØ127                     =            0.                     0
   16       DINØ128                     =            0.                     0
   17       DINØ129                     =            0.                     0
   18       DINØ130                     =            0.                     0
   19       DINØ131                     =            0.                     0
   20       DINØ132                     =            0.                     0
   21       DINØ133                     =            0.                     0
   22       DINØ134                     =            0.                     0
   23       DINØ135                     =            0.                     0
   24       DINØ136                     =            1.                     1
   25       DINØ137                     =            1.                     1
   26       DINØ138                     =            0.                     2
   27       DINØ139                     =            0.                     2
   28       DINØ153                     =            1.                     5
   29       DINØ154                     =            1.                     5
   30       DINØ155                     =            1.                     5
   31       DINØ156                     =            1.                     0
   32       DINØ157                     =            1.                     0
   33       DINØ158                     =            1.                     0
   34       DINØ159                     =            1.                     3
   35       DINØ160                     =            1.                     0
```

Figure 5-4. Translated simulation state list printout.

APPENDIX A

PROGRAM FLOW CHARTS

PRECEDING PAGE BLANK NOT FILMED.

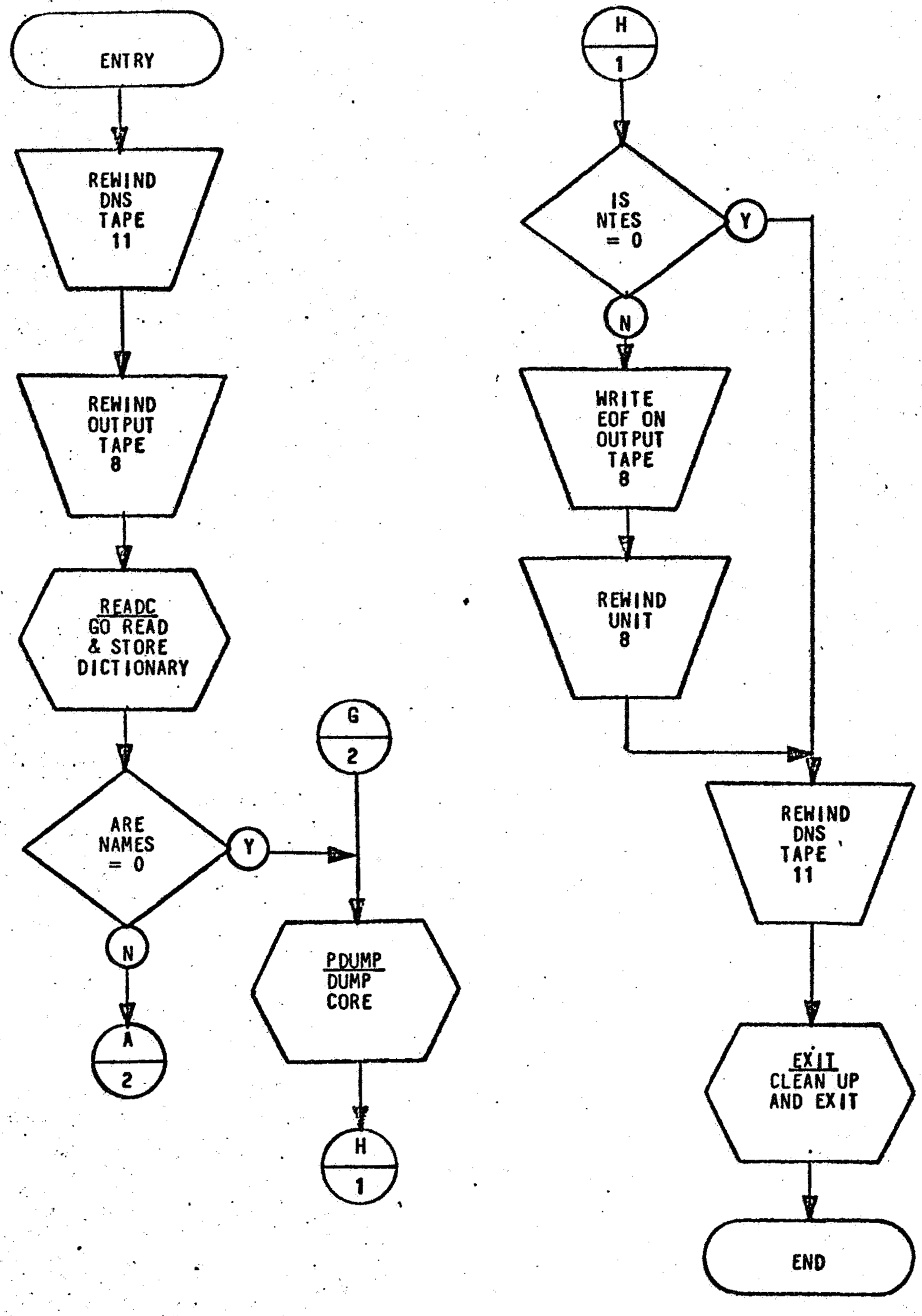


Figure A-1. SUBROUTINE DNSINP (1 of 4)

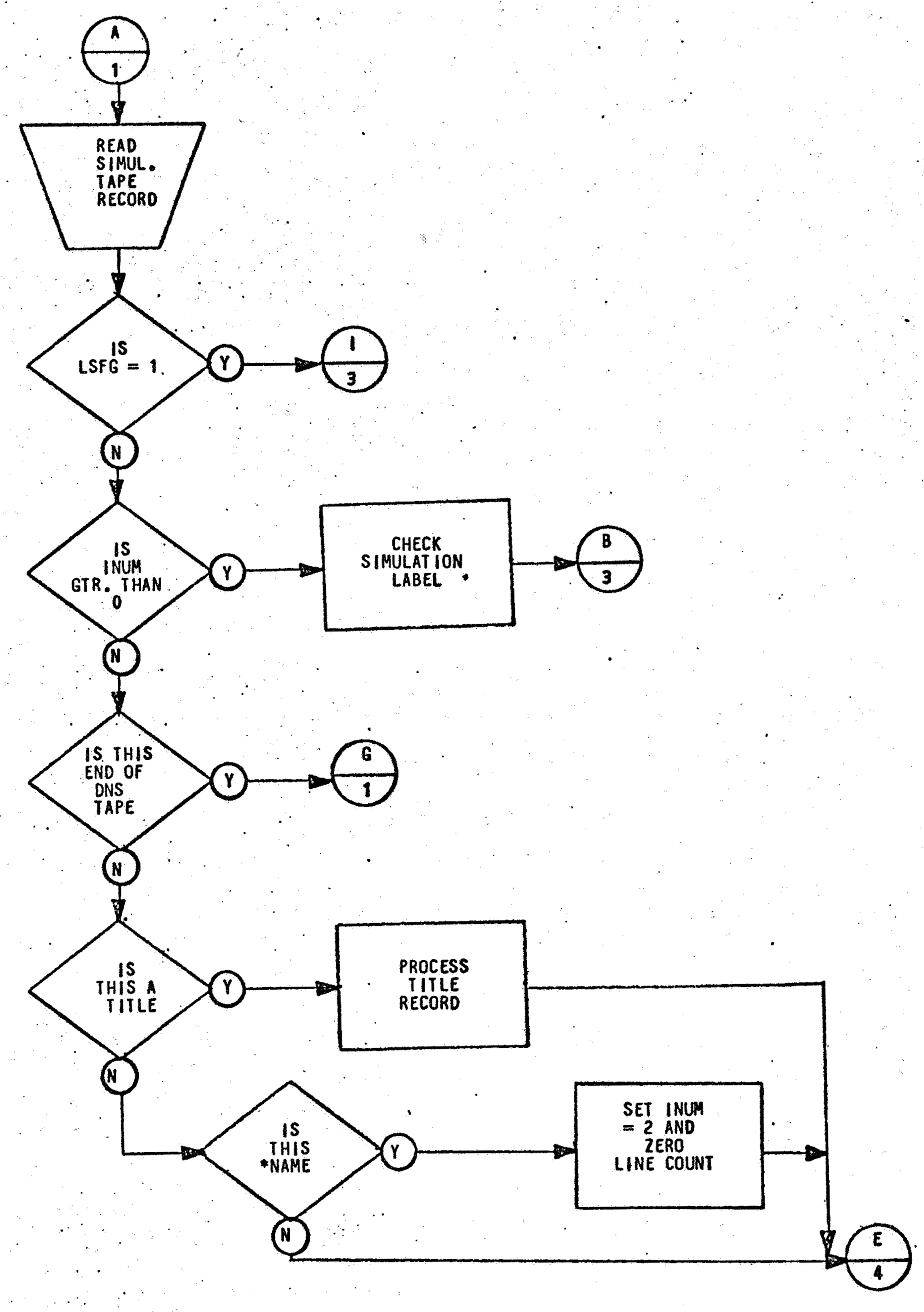


Figure A-1. SUBROUTINE DNSINP (2 of 4)

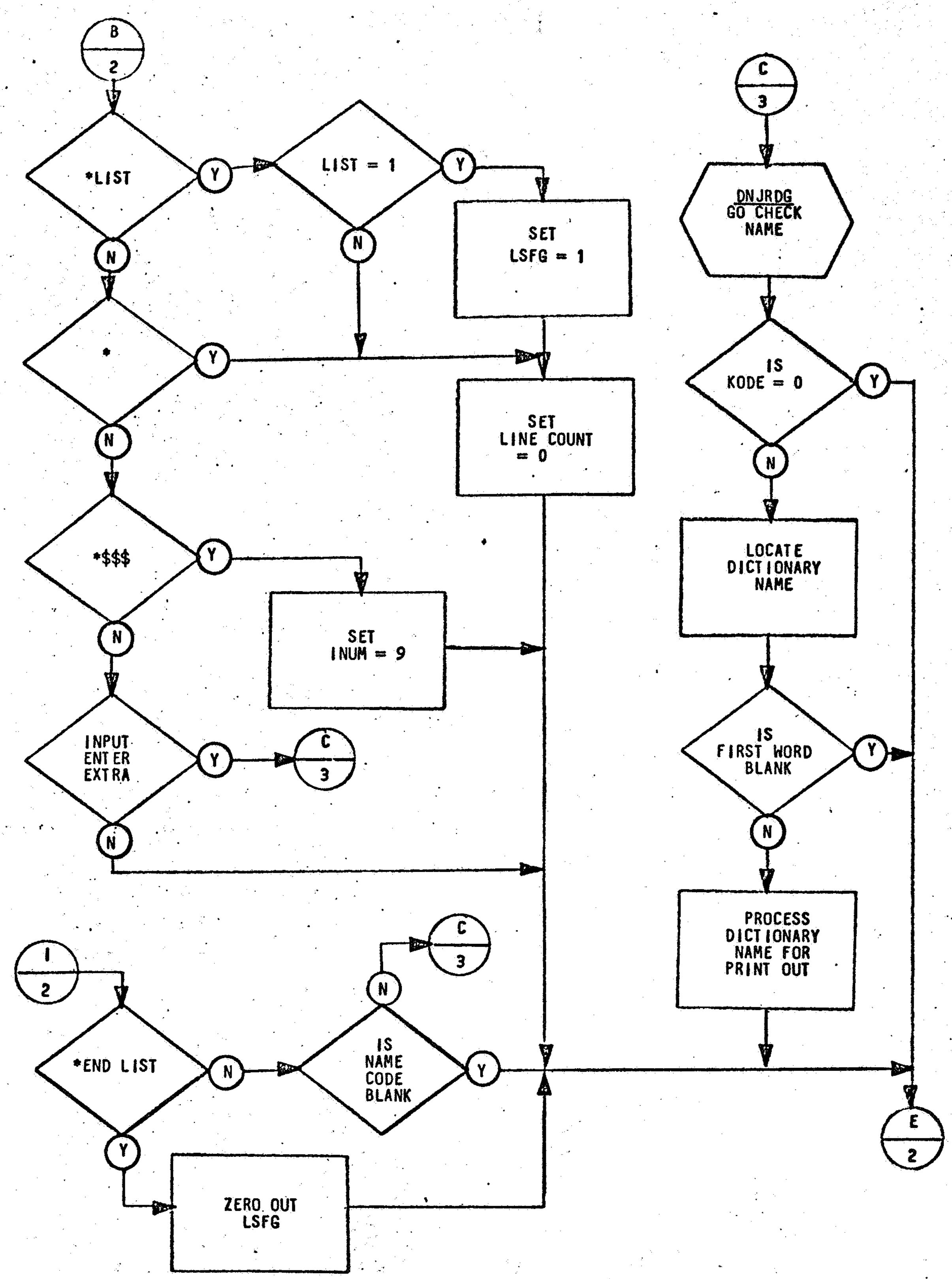


Figure A-1. SUBROUTINE DNSINP (3 of 4)

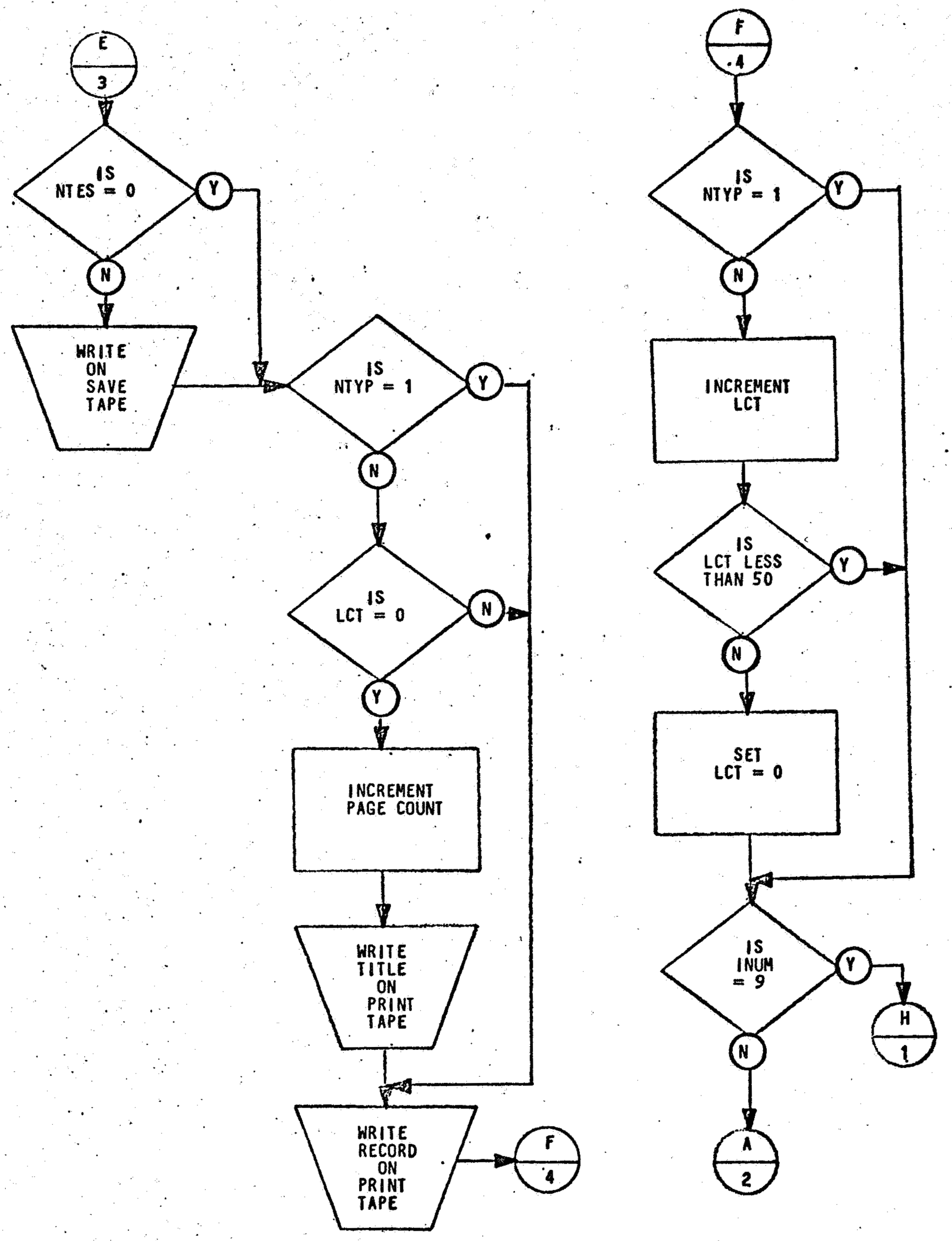


Figure A-1. SUBROUTINE DNSINP (4 of 4)

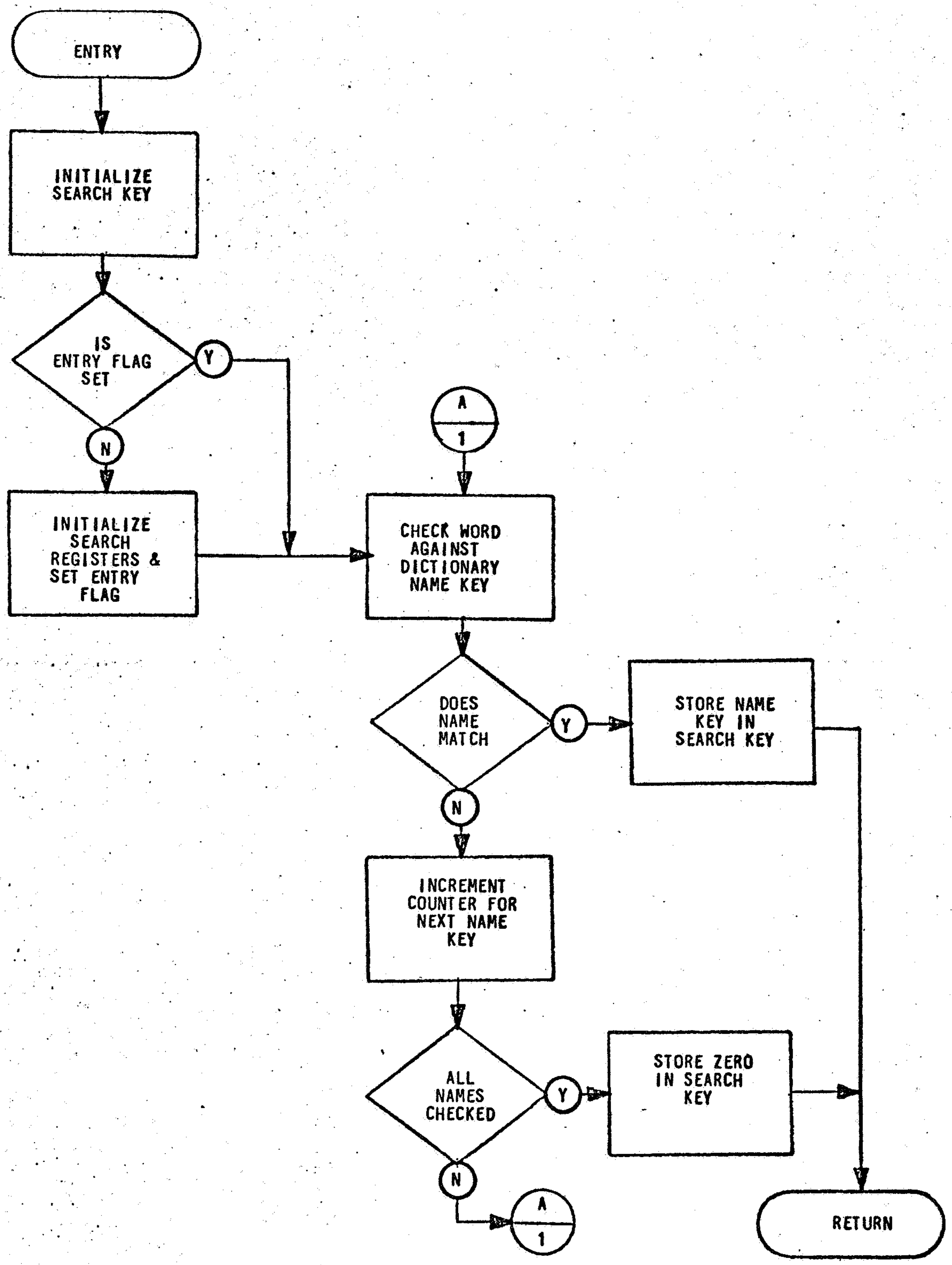

ENTRY
INITIALIZE SEARCH KEY
IS ENTRY FLAG SET
Y
N
INITIALIZE SEARCH REGISTERS & SET ENTRY FLAG
A
1
CHECK WORD AGAINST DICTIONARY NAME KEY
DOES NAME MATCH
Y
STORE NAME KEY IN SEARCH KEY
N
INCREMENT COUNTER FOR NEXT NAME KEY
ALL NAMES CHECKED
Y
STORE ZERO IN SEARCH KEY
N
A
1
RETURN

Figure A-2. SUBROUTINE DNSRDG (1 of 1)

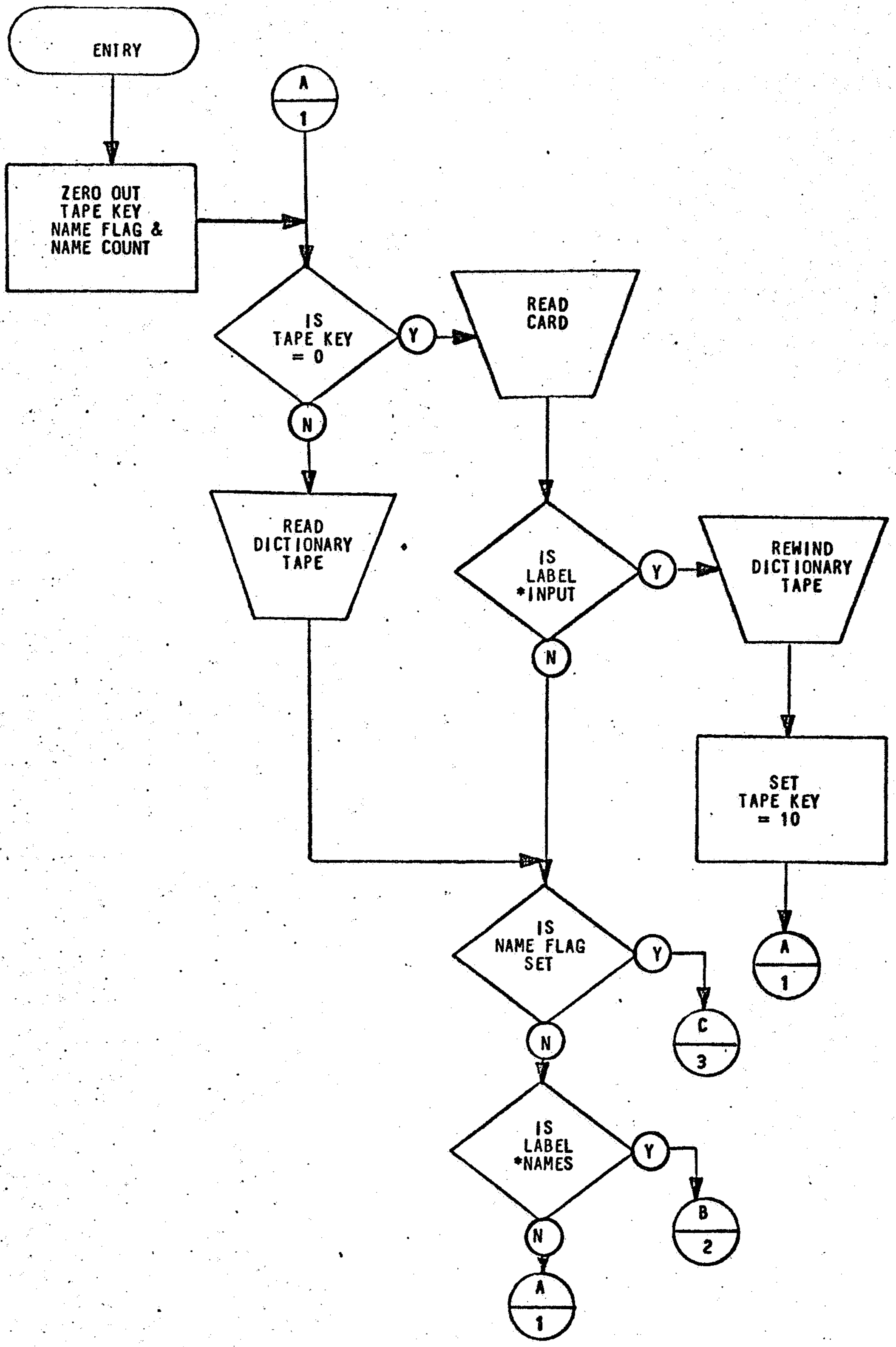


Figure A-3. SUBROUTINE READC (1 of 3)

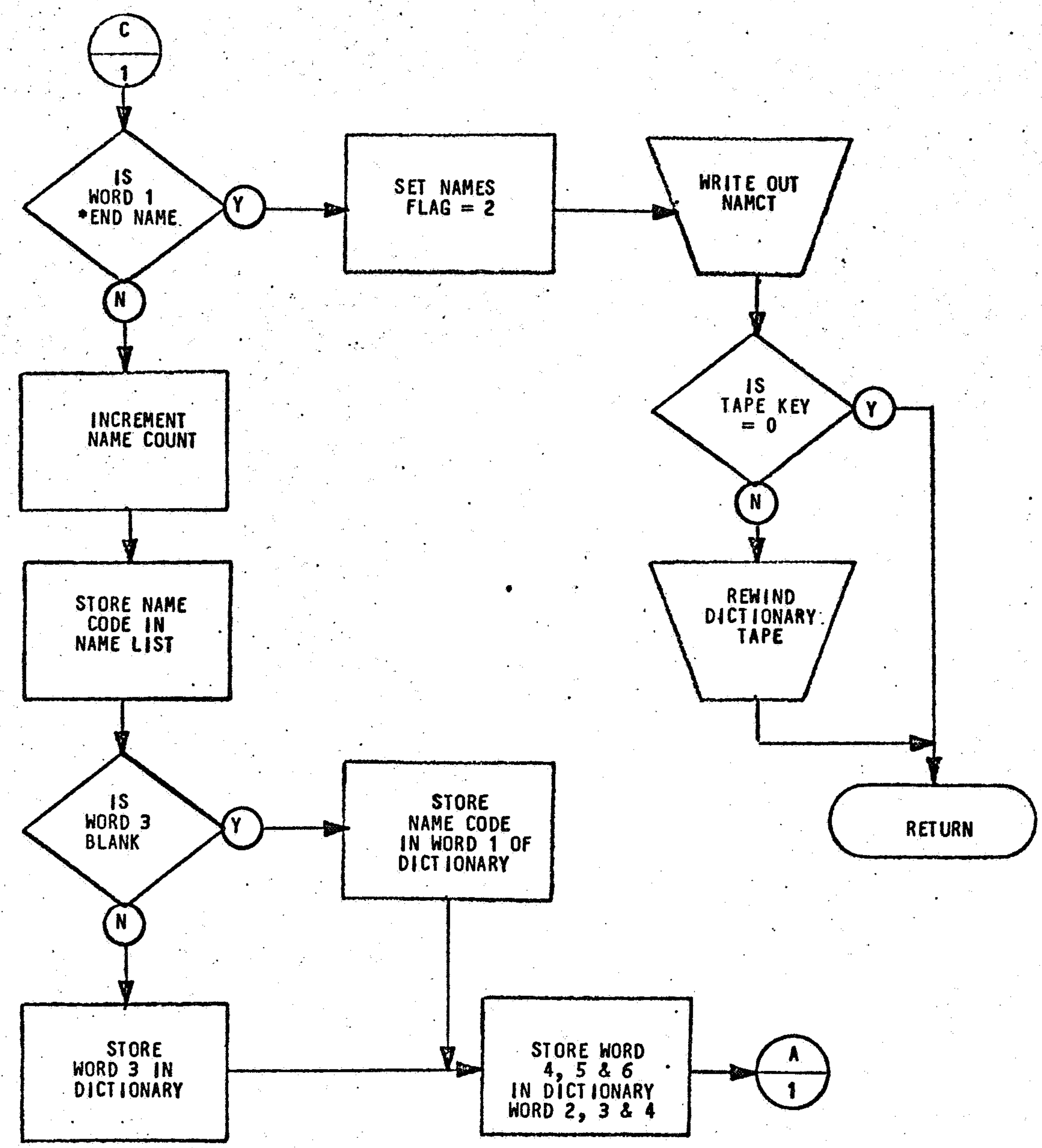

C
1
IS WORD 1 *END NAME
Y
N
SET NAMES FLAG = 2
WRITE OUT NAMCT
IS TAPE KEY = 0
Y
N
REWIND DICTIONARY TAPE
RETURN
INCREMENT NAME COUNT
STORE NAME CODE IN NAME LIST
IS WORD 3 BLANK
Y
N
STORE NAME CODE IN WORD 1 OF DICTIONARY
STORE WORD 3 IN DICTIONARY
STORE WORD 4, 5 & 6 IN DICTIONARY WORD 2, 3 & 4
A
1

Figure A-3. SUBROUTINE READC (2 of 3)

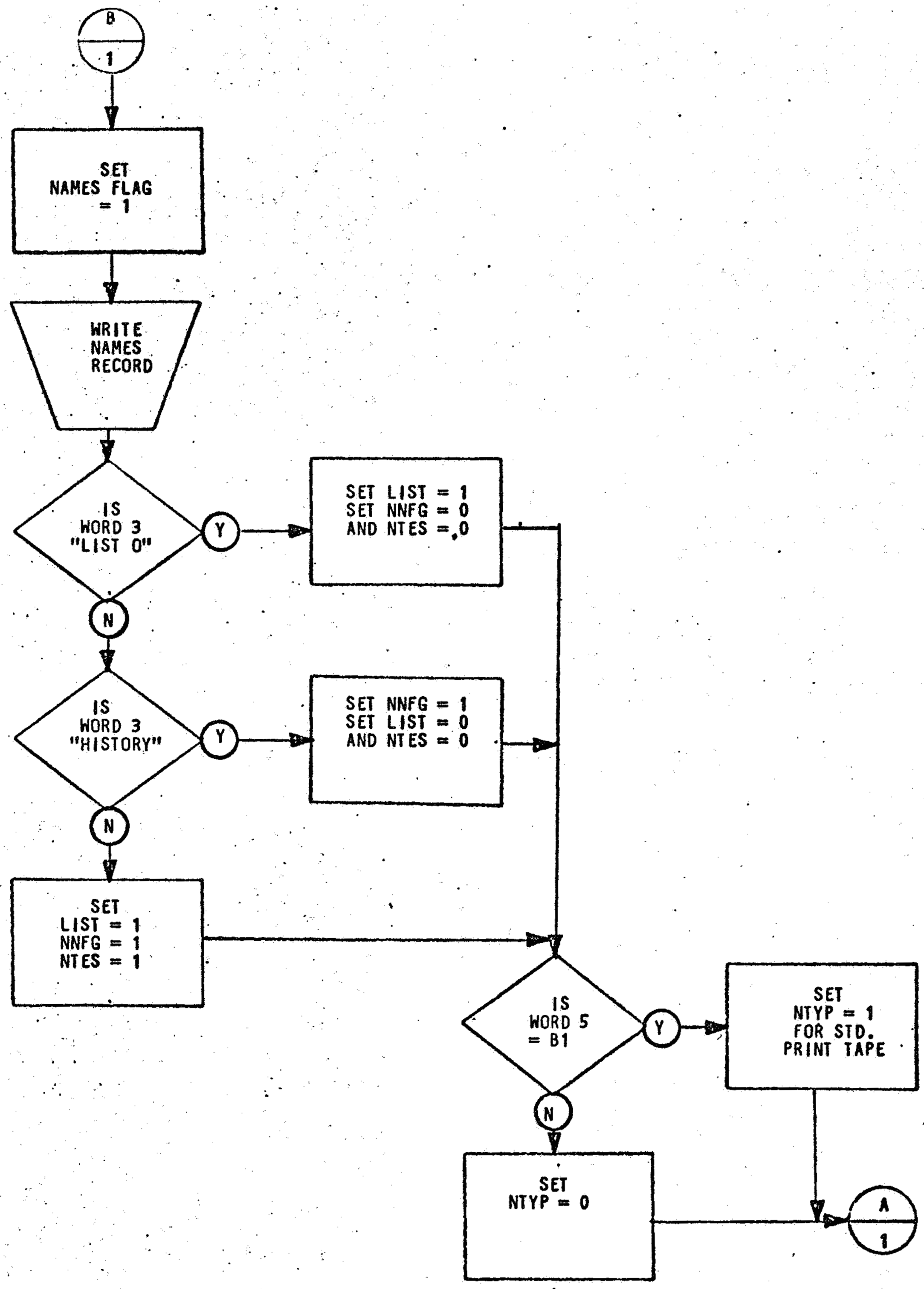

B
1
SET NAMES FLAG = 1
WRITE NAMES RECORD
IS WORD 3 "LIST 0"
Y
SET LIST = 1 SET NNFG = 0 AND NTES = 0
N
IS WORD 3 "HISTORY"
Y
SET NNFG = 1 SET LIST = 0 AND NTES = 0
N
SET LIST = 1 NNFG = 1 NTES = 1
IS WORD 5 = B1
Y
SET NTYP = 1 FOR STD. PRINT TAPE
N
SET NTYP = 0
A
1

Figure A-3. SUBROUTINE READC (3 of 3)

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLES

The following is an alphabetical listing of the terms used in the DNS Translator Program.

NAME	DESCRIPTION
IN	Input name counter.
INONE	Entry flag.
INUM	Translation flag.
KARD	Data input buffer.
KODE	Name code.
LCT	Line counter.
LIST	Simulation list flag.
LSFG	List processing flag.
NAMCT	Name count.
NAME	Simulation name.
NBUF	Spare word.
NNFG	Simulation history flag.
NPAG	Page counter.
NSWDS	Simulation I/O buffer.
NTAPE	Input tape flag.
NTES	Output tape flag.
NTRAN	Dictionary name.
NTYP	Input tape flag.

2.0 DEFINITIONS

NAME	DESCRIPTION
IN	A word in subroutine READC used as a counter to store the number of names read when processing the Dictionary data (cards or tape) supplied for the run.

NAME	DESCRIPTION
INONE	A word in subroutine DNSRDG which is set during the first entry to signal subsequent entries that initialization of the name search routines has been accomplished.
INUM	A word in subroutine DNSINP used as a flag to signal that translation of a simulation output tape is in progress. The flag is set to 2 when the simulation *NAME label is encountered, and is in turn set to 9 when the end of simulation *$$$$$ label is encountered.
KARD	A 15 word input buffer containing a 14 word array KARD and an end buffer word NBUF. It is used in subroutine READC for temporary storage when processing the program data and dictionary cards or tape.
KODE	A word in common block /NTEST/ used as a signal and to store the reference position at which an encountered simulation name was located in the dictionary list. If the simulation name was not included in the dictionary list, the value is set to zero.
LCT	A word in subroutine DNSINP used as a coûnter for storing the number of print lines processed. It is tested before writing, and if the count has been zeroed, paging and titling instructions are activated. It is incremented and tested after writing, and when the count equals 50, LCT is rezeroed. It is also zeroed if specified Simulation Control Labels are encountered.
LIST	A word in common block /NAMEL/ used as a flag to signal whether names in the simulation state lists are to be translated. The flag is set to 1 or 0 in subroutine READC when the *NAMES card is read, and is later used by subroutine DNSINP to select an appropriate processing path. It is set to 1 if the third word on the *NAMES card is blank or contains a "List 0" label. If not, it is set to 0. If set to 1, the state list is to be translated: If 0, it is to be copied exactly as it is.

NAME	DESCRIPTION
LSFG	A word in subroutine DNSINP used as a flag to signal that a Simulation State List is being processed. It is set when a Simulation "*LIST" Label is encountered if the LIST flag has also been set, and is zeroed when a succeeding *END LIST Label is encountered.
NAMCT	A word in common block /NAMEL/ used to store the number of names which were contained in the Dictionary (cards or tape) used for the run. Its value is set in subroutine READC from the final value of counter IN, and is used in subroutine DNSRDG when initializing name search routines.
NAME	A 4000 word array in common block /NAMEL/ used to store the list of Names being defined in the Dictionary. The array is used in conjunction with counter IN in subroutine READC when the Dictionary data is read in. Each Name is stored in BCD format in a consecutive position in the array, and its associated 4 word definition (translation name) is stored in array NTRAN.
NAMEL	A 20,003 word common block reserved for storing data associated with processing of Dictionary names, and translation names or definitions. It contains LIST, NNFG, NAMCT, a 4000 word array NAME, and a 16,000 word array NTRAN.
NBUF	A word in common block /KARD/ included for future use when using Dictionary data tapes having a 15th data word.
NNFG	A word in common block /NAMEL/ used as a flag to signal whether names in the simulation history (action and reaction table printout) are to be translated. The flag is set in subroutine READC when the control cards are read, and is later used by subroutine DNSINP to select an appropriate processing path. It is set to 1 if word 3 of the *NAMES card is blank, or contains the label HISTOR. If word 3 contains LIST0, the flag is set to 0. If set to 1,

NAME	DESCRIPTION
	the histories are to be translated. If set to 0, the histories are merely to be copied.
NPAG	A word in subroutine DNSINP used as a counter for storing the page number to be assigned to each page of the program printout.
NSWDS	A 20 word input buffer used in subroutine DNSINP for temporary storage of a Simulation tape record, used in subroutine DNSRDG when conducting the name search, and again used in DNSINP as the Output Print buffer.
NTAPE	A word in subroutine READC used as a flag to signal that the Input Dictionary is to be read from a special Input tape. It is set to 1 if the first data card is labeled *INPUT, and all further Dictionary data will be read from the special Input. If 0, the data is read from the standard load tape.
NTES	A word in common block /NTEST/ used as a flag to signal that the program results are to be stored on an output save tape. It is set to 1 in subroutine READC if word 3 of the *NAMES card is blank. If not, the flag is zeroed. The flag is used by subroutine DNSINP to select an appropriate process ing path.
NTEST	A common block of 3 locations containing KODE, NTES, and NTYP.
NTRAN	A 16,000 word array in common block /NAMEL/ used for storing the 4 word (up to 24 BCD characters maximum) definitions or translation names associated with each name contained in array NAME. A word in common block /NTEST/ used as a flag to signal whether a Simulation Output Tape or the standard system printout is to be translated. The flag is set to 1 in subroutine READC if the 'B1' label is encountered on the control card. If not, the flag is zeroed. The flag is tested in subroutine DNSINP to select the appropriate processing path.

SECTION

6

DTC/REFTAB PROGRAM

CONVAIR DIVISION OF GENERAL DYNAMICS CORPORATION

6

DTC/REFTAB PROGRAM

AUTHOR:	A. R. Stone Convair division of General Dynamics

PURPOSE:

The DTC REFTAB Program was written to provide a printout of variable names and reference data contained on the AMA Model tape generated by the Down Translation and Culling Program. The printed data contains essential information on inactive variables in the model, and is useful to the model builder in validating the AMA Model. The AMA Model tape contains a reference table wherein each variable has been classified as an Initiator, Transactor or Terminal and assigned a failure analysis candidacy. This program will read tne DTC AMA tape, locate the equation and reference tables, convert the data to BCD and print a listing of the variable number, three letter code, engineering name, variable type and failure classification.

RESTRICTIONS:

1. The program must run on an IBM 7090/94 with IBJOB systems capability.

2. The program must be used with an AMA Model tape previously created by the DTC Program. The model cannot contain more than 4000 variables.

STORAGE:

The program occupies $13,358)_{10}$ consecutive core locations beginning at $03004)_8$ and ending at $35062)_8$ and consists of the following seven sub-programs.

1.	CONTRD	Driver
2.	DTCINZ	Tape data location and read in
3.	ERRMEX	Error message printing
4.	MANIZ	Data formatting for print

5.	PBCDZ	Conversion from BIN to BCD
6.	PRINZ	Printing of converted data
7.	TRANL	3 letter variable code conversion

USE:

The program does not require any control cards and is used as shown in following example of deck setup.

```
$JOB
$PAUSE
$ATTACH        B6
$AS            SYSUT6
$EXECUTE       IBJOB
$IBJOB         GO, MAP
Binary program deck
7/8 (EOF)
```

DTC AMA B7 save tape is loaded as a B6 input as shown in proceeding example. The output consists of B1 printout of information extracted from the tape.

METHOD:

The program first sets up a Processing Director Flag (PDF) to a value of one (1).

1. Sub-program DTCNP is called to locate and read in the reference table prologue and names totals. The tape is first rewound to insure start and load point. A counter (KRY) is incremented as each record is read. KRY is not checked at this point in the processing, but was included here for future incorporation record counts and error exits. The first word of each record contains the number of items of data contained in the record. The second word contains the total number of words required to store these data items in the record. Each record is read in the following order: first word NITEMS then second word NWORDS then rest of record into KBUF, with the value found in the second word used as the number of the words to be read into array KBUF. The first word of KBUF is then tested to see if it contains the label *REFER. When *REFER is located, flag KTRAK is set to value of 1 signalling that the record has been located. The values found in the 18th, 19th, 20th words of KBUF are transferred into common locations ACTCT, INACT and NAMCT respectively for subsequent use by the program, and program control is returned to the driver program. The flag KTRAK is tested for value of 1 (flag was set to 1 when the reference prologue record was located). If value is not 1, a transfer to error print will ensue.

2. Control is returned to sub-program DTCINP with the PDF equal to 2 to signal that the reference table is to be read. The reference table record immediately follows the reference prologue, and consists of one continuous record which is read and stored in array KREF. The buffer cells NITEMS and NWORDS are zeroed and the tape is rewound. KTRAK is set to a value of 2 and control is returned to the driver sub-program. Testing of KTRAK is repeated with same results upon error (value other than 2).

3. The program then returns to DTCINP with the PDF set to a value of 3 and the tape records are searched for the word *NAMES. The NAMES record should be the first record on the tape so counter KRY is tested and if it exceeds value of one, KTRAK is set to seven and an error return to the driver sub-program is initiated. When the names prologue record is found and read into KBUF, flag KTRAK is set to value of 3 and the word per record buffer is zeroed. Control again transfers to the driver sub-program where KTRAK is tested as before. If KTRAK = 3 the PRINT routine is called with PDF set to a value of 1. A limit value 'M' is set to 1 and the contents of ACTCT are tested. If ACTCT is found to be zero the value of 'M' is increased to 2. If value of ACTCT is not zero, value of 'M' remains 1. Paging data is initialized and control is returned to the main driver sub-program. DTCINP is again called with the PDF now at a value of four.

4. A names record is read into KBUF. The format of the names record is: word one is the total number of words in the record, word two is the number of items (names) described in record, word three is number of words in the record describing names. Each name is grouped in three to seven word sections. The first word of the section contains the numerical code number of the name. Word two will contain a number from 1 to 5 stating how many words are used in the variable name. The rest of the words in the section will contain the actual variable name in BCD. There will be as many sections as shown in word two of the record. The names are transferred individually from KBUF into array KTAB. As a word is transferred to KTAB, it is tested for a single dollar sign signalling the end of the names. When the dollar sign

is encountered, KTRAK is set to a value of 6 to signal that the last name has been read. The names are placed in a nine by NITEMS array with each name being placed in words three through seven of each row of the array. Each row of the array is given an ascending sequence number which is extracted from the first word of each name section in the names record. (The second word of each row is reserved for the 3 letter coded form of the name. Word eight will contain the type classification and word nine will have the zeros and ones failure candidacy classification). When the names are placed into the matrix KTAB five words are used for each name regardless of number actually in the name. The words not containing data are filled with blanks. A count of variables read in is maintained in KNT and is compared with that value stored in ACTCT, flag KTRAK is set to a value of 4 if name count (KNT) is less or equal to the number of active variables (value in ACTCT). If the current name total is more than the ACTCT total, KTRAK is set to five to indicate that the inactive variables are now in process. Control is transferred back to the driver sub-program when the current names record has been stored.

5. Control is transferred to subroutine MANIP where the current data stored in the matrix KTAB is converted to BCD for printout. The first word of a matrix row is loaded into the accumlator and reduced by one. This number represents the numerical code number of the variable, and will be converted to a coded three letter name for the variable. To determine the exact three letters the program transfers to TRANS. TRANS divides the code number by 26 squared (676), the remainder by 26, and its remainder by 1. The numbers obtained are used as 6 bit search keys for a table in core storage. The table is searched 6 bits at a time and each 6 bits replaced by the octal value found in the table. When the number has been completely converted to a three letter alpha code, control is returned to subroutine MANIP where RH zeroes in the code are replaced with blanks and the word is stored in word two of its row in KTAB. Word one of the row (code number of the variable) is converted to BCD in subroutine PBCD. The failure code classifications are now extracted from the reference table data previously stored in array KREF and placed in the ninth word of the row in the matrix KTAB. Finally

the coded classification for variable type is extracted from array KREF and converted to a BCD word which in turn is placed in the eighth word of the row in the matrix KTAB. When reference data for all names in the current names record have been processed, the program transfers to subroutine PRINT.

6. Printing is accomplished directly from the array KTAB one row at a time with spacing to separate data. As each row is printed a counter is incremented and compared with the value stored in ACTCT which represents the count of active variables in the model. When the count and ACTCT are equal, the paging is set back to 1 and a new page is selected and the inactive variables are printed, if any are in the model. This is determined by a test of the storage cell INACT which contains the count of inactive variables in the model. When NITEMS names from the names record have been processed, the value of KTRAK is checked. If KTRAK is 4 or 5, the program transfers to 4 to process another names record. If KTRAK is 6 the program cleans up and exits.

OUTPUT FORMAT: Figure 6-1 is a sample of a test model that was processed by the DT&C program for use in demonstrating the REFTAB output shown in Figure 6-2. The subtitles listed under the title represent columns in the two dimensional array KTAB. Each row is numbered under subtitle NO and is the first word in each row of KTAB. This number is the numerical code number of the variable in the DNS model. Under CODE the three letter code that will be used in the DNS program printouts is listed. The three letter code will correspond to the code number of the variable. Variable #1 is AAA and Variable #26 is AAZ and Variable #17576 would be ZZZ. The full engineering names of the variables are printed in up to five words (thirty characters maximum). Spaces not containing data will be filled with blanks. In the example, a portion of the variables are named in such a fashion as to provide a test reference of the output of the REFTAB program. Variable #14 (AAN) is named TERM-BOTH and in this instance has no other meaning other than to provide an example to show that the program has determined it to be a terminal and is a failure candidate for both zeros and ones. The V. TYPE (variable type) lists the classification for each variable. TERM-L is a shortened version of Terminal, as is TRANS for Transactor, and INIT-R for

```
                        DISCRETE NETWORK SIMULATOR-DOWNTRANSLATED TIME CARDS              PAGE   1
 AAA   DI1345                             S   0S   0S   0S   0S   0S   0AIFU  I0
 AAB   NODE115A3A1J34P                    S   0S   0S   0S   0S   0S   0AD X  X0
 AAC   PBUS5A31D111                       S   0S   0S   0S   0S   0S   0AI I  B0
 AAD   PBUS115A3A21D111                   S   0S   0S   0S   0S   0S   0AI I  B0
 AAE   COIL115A3A1K40J15PPSZ              S   0S   0S   0S   0S   0S   0AIFU  K0
 AAF   CONT115A3A1K40J15AABB              S   4S   5S   6S   4S   5S   6ADUM  C0
 AAG   CONT115A3A2K51J20DDEE              S   4S   5S   6S   4S   5S   6ADUM  C0
 AAH   CK1N115A3A1J34P                    S   0S   0S   0S   0S   0S   0AIUS  Y0
 AAI   CK2N115A3A1J34P                    S   0S   0S   0S   0S   0S   0AIUS  Y0
 AAJ   PRSW5A350A45C                      S   0S   0S   0S   0S   0S   0AI I  S0
 AAK   PRSW5A350A45C                      S   0S   0S   0S   0S   0S   0AI I  S0
 AAL   SRTERM                             S   0S   0S   0S   0S   0S   0AIFU  V0
 AAM   RESET                              S   0S   0S   0S   0S   0S   0AI  I B0
 AAN   TERM-BØTH                          S   0S   0S   0S   0S   0S   0AIFU  I0
 AAØ   SELFREF                            S   0S   0S   0S   0S   0S   0A  I  G1
 AAP   INIT                               S   0S   0S   0S   0S   0S   0A  I  G1
 AAQ   TRAN                               S   0S   0S   0S   0S   0S   0A  I  G1
 AAR   TRAN-ØNE                           S   0S   0S   0S   0S   0S   0ADFU  P0
 AAS   TRAN-BØTH                          S   0S   0S   0S   0S   0S   0AIFU  I0
 AAT   INIT-ØNE                           S   0S   0S   0S   0S   0S   0ADFU  P0
 AAU   INIT-BØTH                          S  13S  15S  17S   5S   6S   7ADUM  C1
 AAV   UNUSEDACT-BØTH                     S  13S  15S  17S   5S   6S   7ADUM  C1
```

Figure 6-1. Time card listing, DTC program.

```
ACTIVE VARIABLES, A M A MØDEL    DT+C PRØGRAM         REFERENCE DATA

 NØ.      CØDE      FULL NAME                          V.TYPE      F.CAND

  1       AAA       DI1345                             TERM-L      BØTH
  2       AAB       NØDE115A3A1J34P                    TRAN-R
  3       AAC       PBUS5A31D111                       INIT-R      BØTH
  4       AAD       PBUS115A3A21D111                   INIT-R      BØTH
  5       AAE       CØIL115A3A1K40J15PPSZ              TRAN-R      BØTH
  6       AAF       CØNT115A3A1K40J15AABB              TRAN-R      BØTH
  7       AAG       CØNT115A3A2K51J20CDEE              INIT-R      BØTH
  8       AAH       PIN5A3J34B                         INIT-R      ØNES
  9       AAI       CK1N115A3A1J34P                    TRAN-R
 10       AAJ       CK2N115A3A1J34P                    TRAN-R
 11       AAK       PRSW5A350A45C                      INIT-R      ØNES
 12       AAL       SRTERM                             SRTERM
 13       AAM       RESET                              INIT-R      BØTH
 14       AAN       TERM-BØTH                          TERM-L      BØTH
 15       AAØ       SELFREF                            SRINIT      BØTH
 16       AAP       INIT                               INIT-R      BØTH
 17       AAQ       TRAN                               TRAN-R      BØTH
 18       AAR       TRAN-ØNE                           TRAN-R      ØNES
 19       AAS       TRAN-BØTH                          TRAN-R      BØTH
 20       AAT       INIT-ØNE                           INIT-R      ØNES
 21       AAU       INIT-BØTH                          INIT-R      BØTH
 22       AAV       UNUSEDACT-BØTH                      ****       BØTH
```

Figure 6-2. Reference table printout.

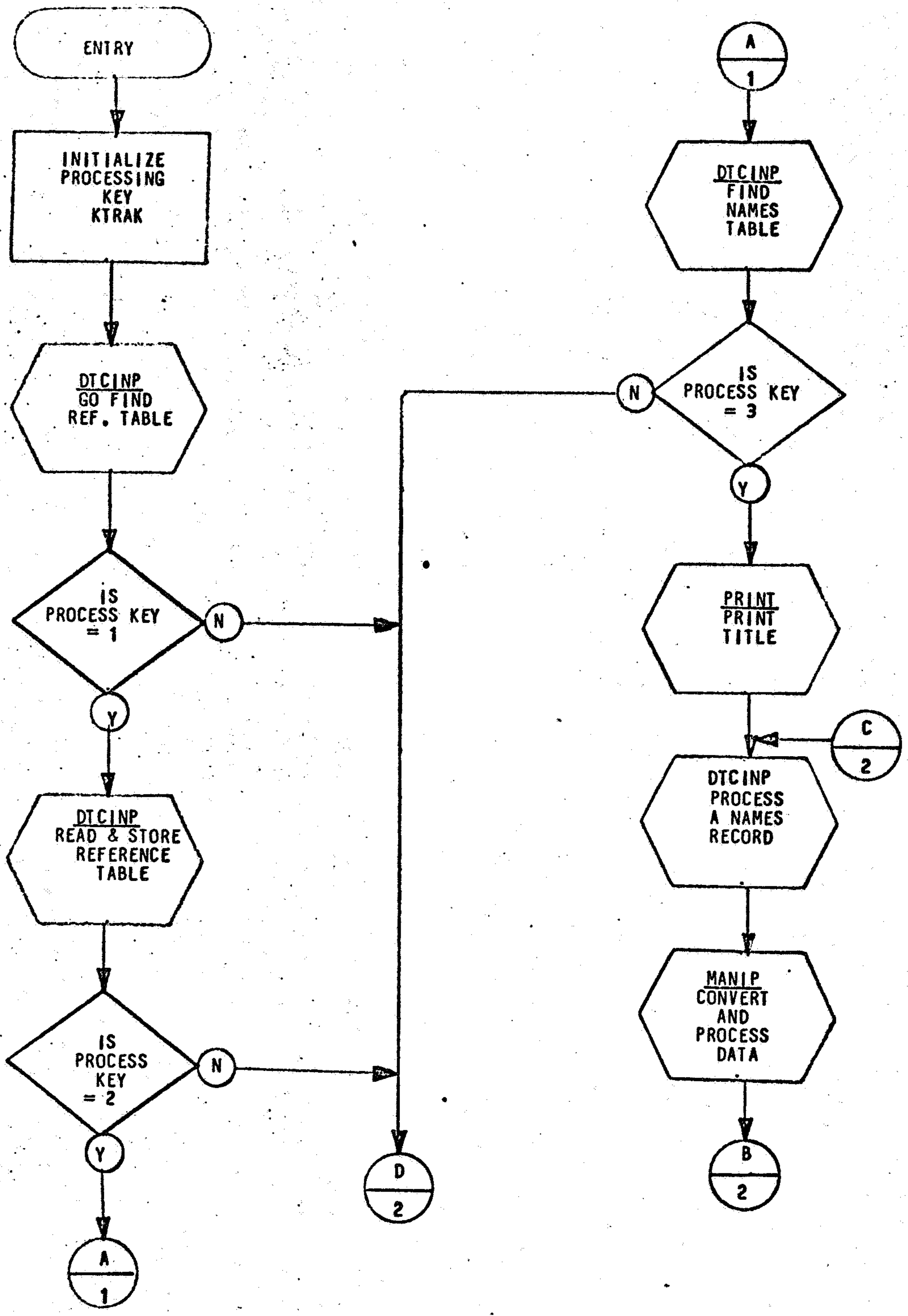
Initiator. The prefix SR indicates that the variable has also been classified as a self referencing variable in addition to other classifications. Variables with four asterisks printed in the type column are variables that have been classified as UNUSED by the DT&C program. Self referencing variables are those that will effect themselves during a simulation. An example would be a spring loaded push button where the push button was equated in such a manner as to turn itself off a certain period of time after an input (initiator) had set its value to one without a second input resetting its value to zero. TERMINALS are variables that will not affect any other variables in the model. They are the ultimate terminus of actions and reactions as a result of inputs (initiators).

The unused variables are often the results of errors in the logic of the model equations or are variables that are actually in the hardware but do not affect nor are affected by any other variables in the model. They may be removed without destroying the validity of the simulation. The failure candidacy is listed under F. CAND and is classified as a failure for either zeros, ones or both for AMA analysis. This classification results from the class code placed in CC79 of the variable time parameter card and the failure code listing on the first two control cards of the DTC data deck. The computer time required for a DT&C run plus a REFTAB run combined are much less than that required for a Prep Editor run, therefore considerable computer time is saved by analysis of the information contained in the listed printout.

APPENDIX A

PROGRAM FLOW CHARTS

PRECEDING PAGE BLANK NOT FILMED.



ENTRY
INITIALIZE PROCESSING KEY KTRAK
DTCINP GO FIND REF. TABLE
IS PROCESS KEY = 1
N
Y
DTCINP READ & STORE REFERENCE TABLE
IS PROCESS KEY = 2
N
Y
A 1
D 2
A 1
DTCINP FIND NAMES TABLE
IS PROCESS KEY = 3
N
Y
PRINT PRINT TITLE
C 2
DTCINP PROCESS A NAMES RECORD
MANIP CONVERT AND PROCESS DATA
B 2

Figure A-1. SUBROUTINE CONTRD (1 of 2)

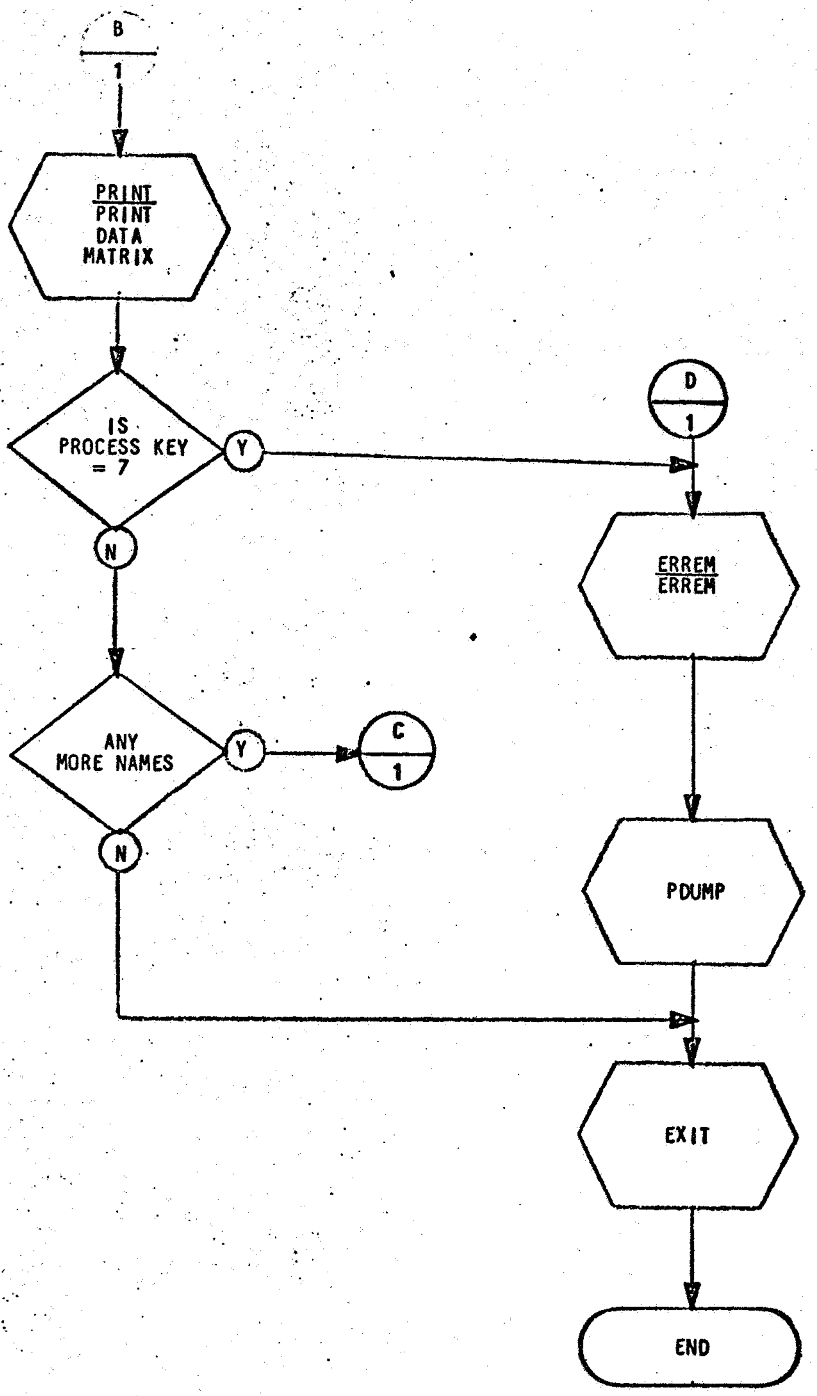


Figure A-1. SUBROUTINE CONTRD (2 of 2)

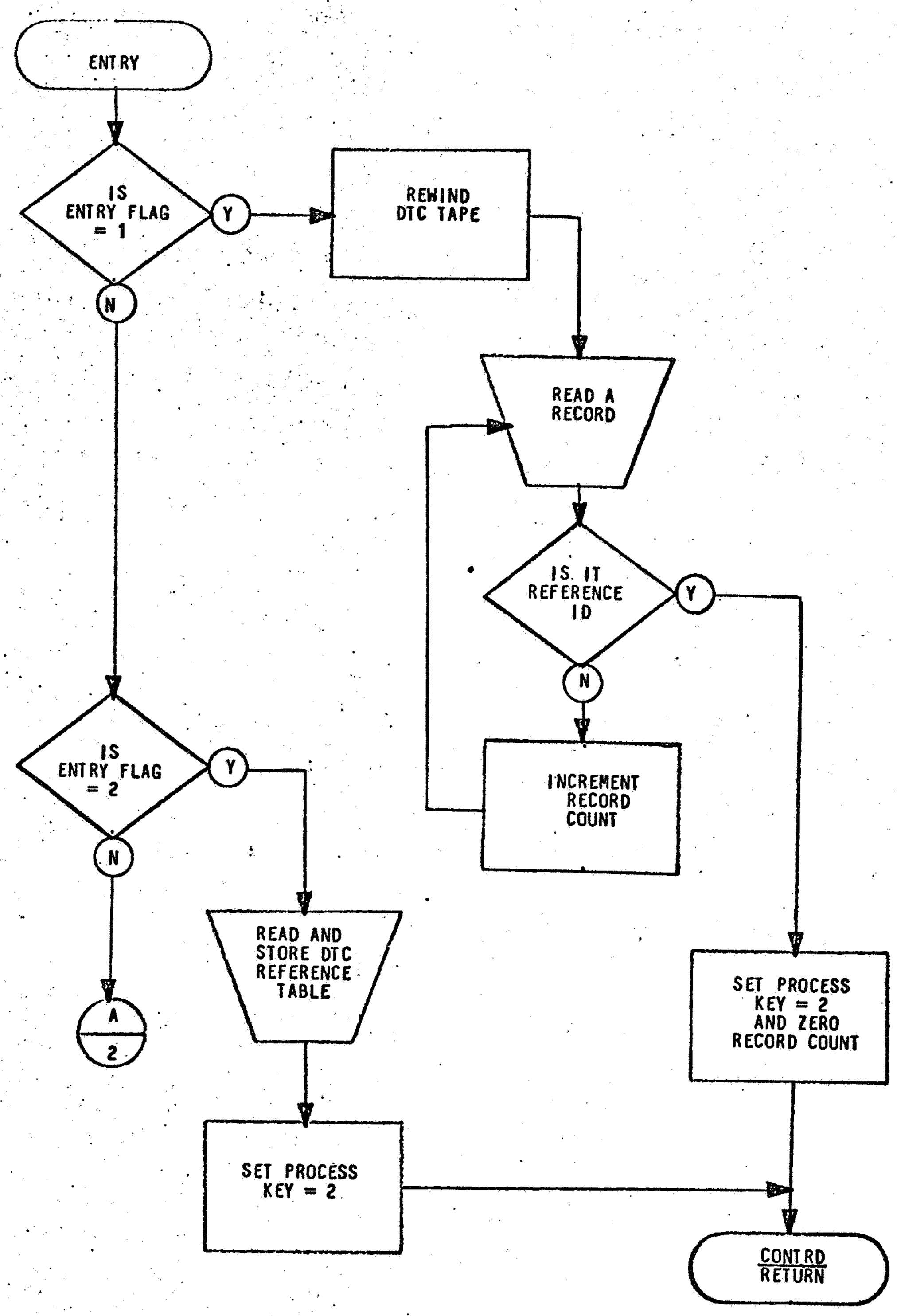


Figure A-2. SUBROUTINE DTCINZ (1 of 3)

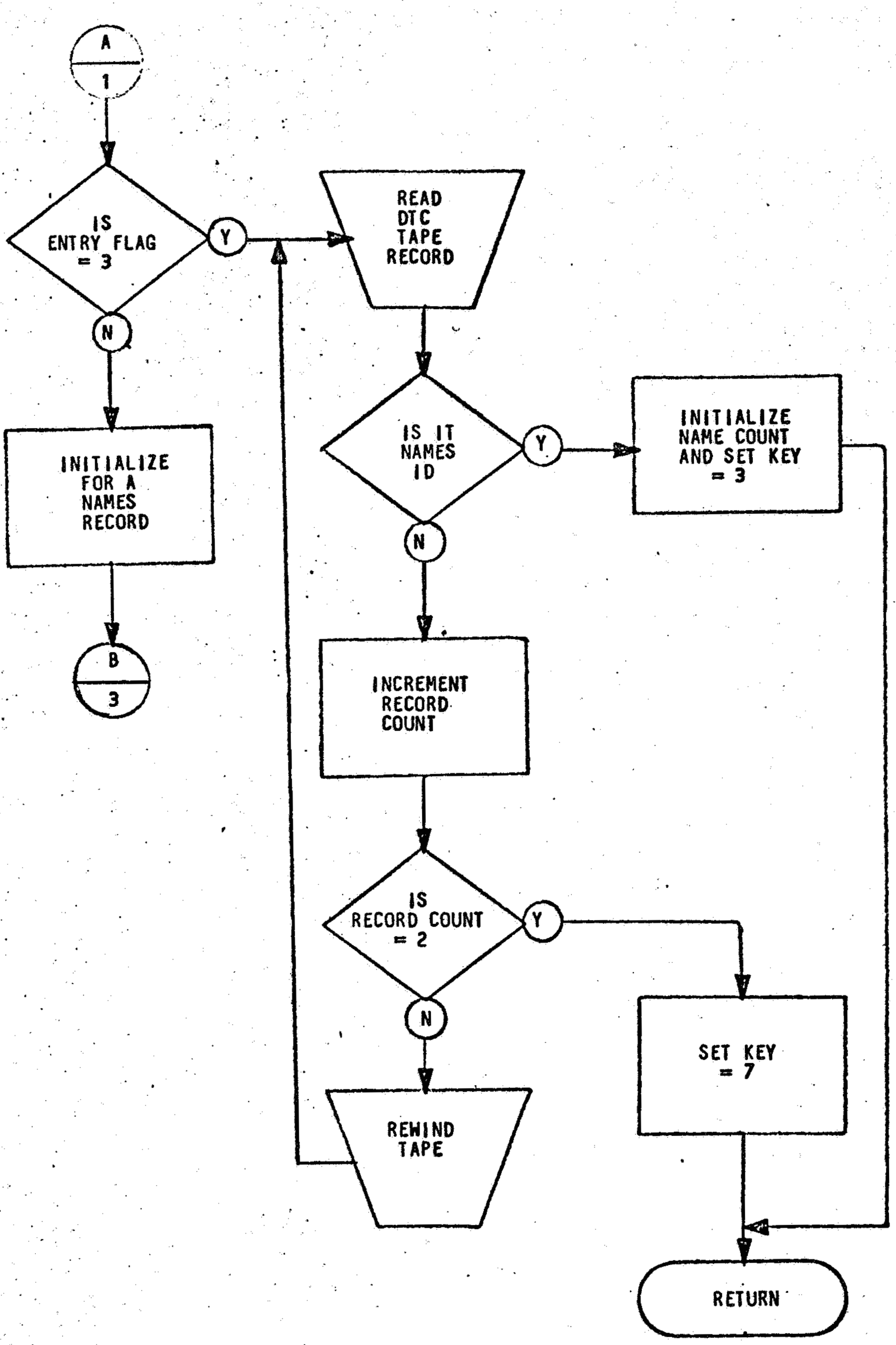


Figure A-2. SUBROUTINE DTCINZ (2 of 3)

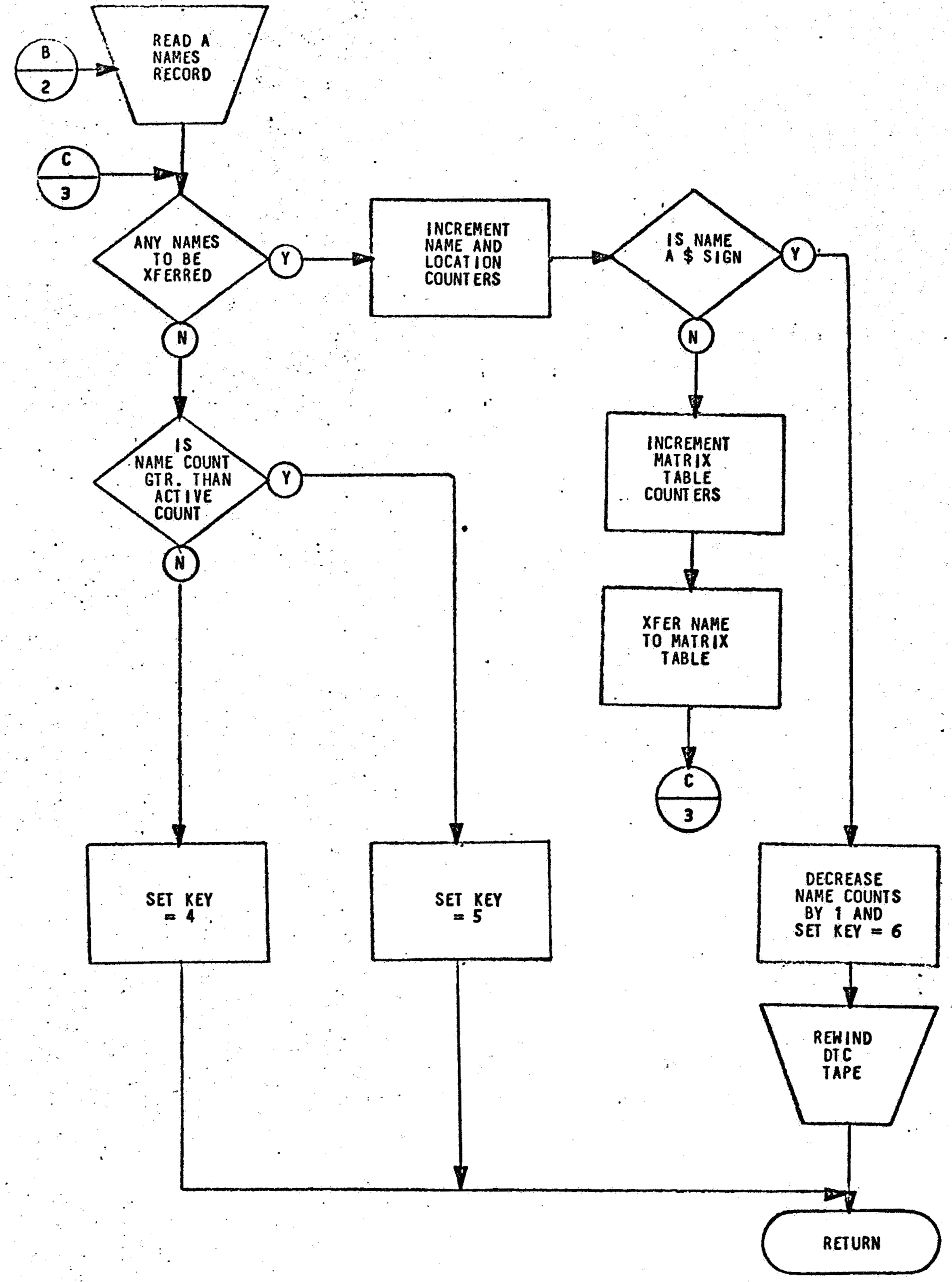

B
2
READ A NAMES RECORD
C
3
ANY NAMES TO BE XFERRED
Y
N
INCREMENT NAME AND LOCATION COUNTERS
IS NAME A $ SIGN
Y
N
IS NAME COUNT GTR. THAN ACTIVE COUNT
Y
N
INCREMENT MATRIX TABLE COUNTERS
XFER NAME TO MATRIX TABLE
C
3
SET KEY = 4
SET KEY = 5
DECREASE NAME COUNTS BY 1 AND SET KEY = 6
REWIND DTC TAPE
RETURN

Figure A-2. SUBROUTINE DTCINZ (3 of 3)

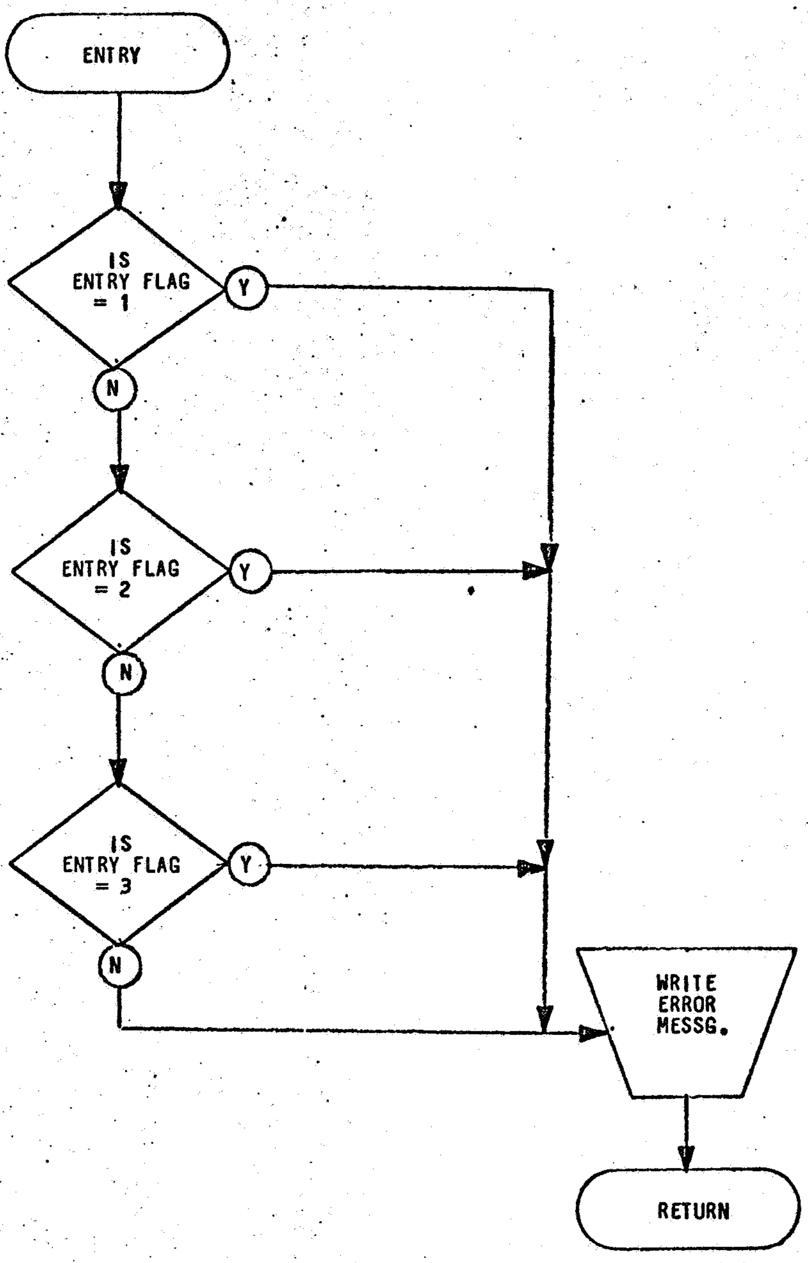


Figure A-3. SUBROUTINE ERRMEX (1 of 1)

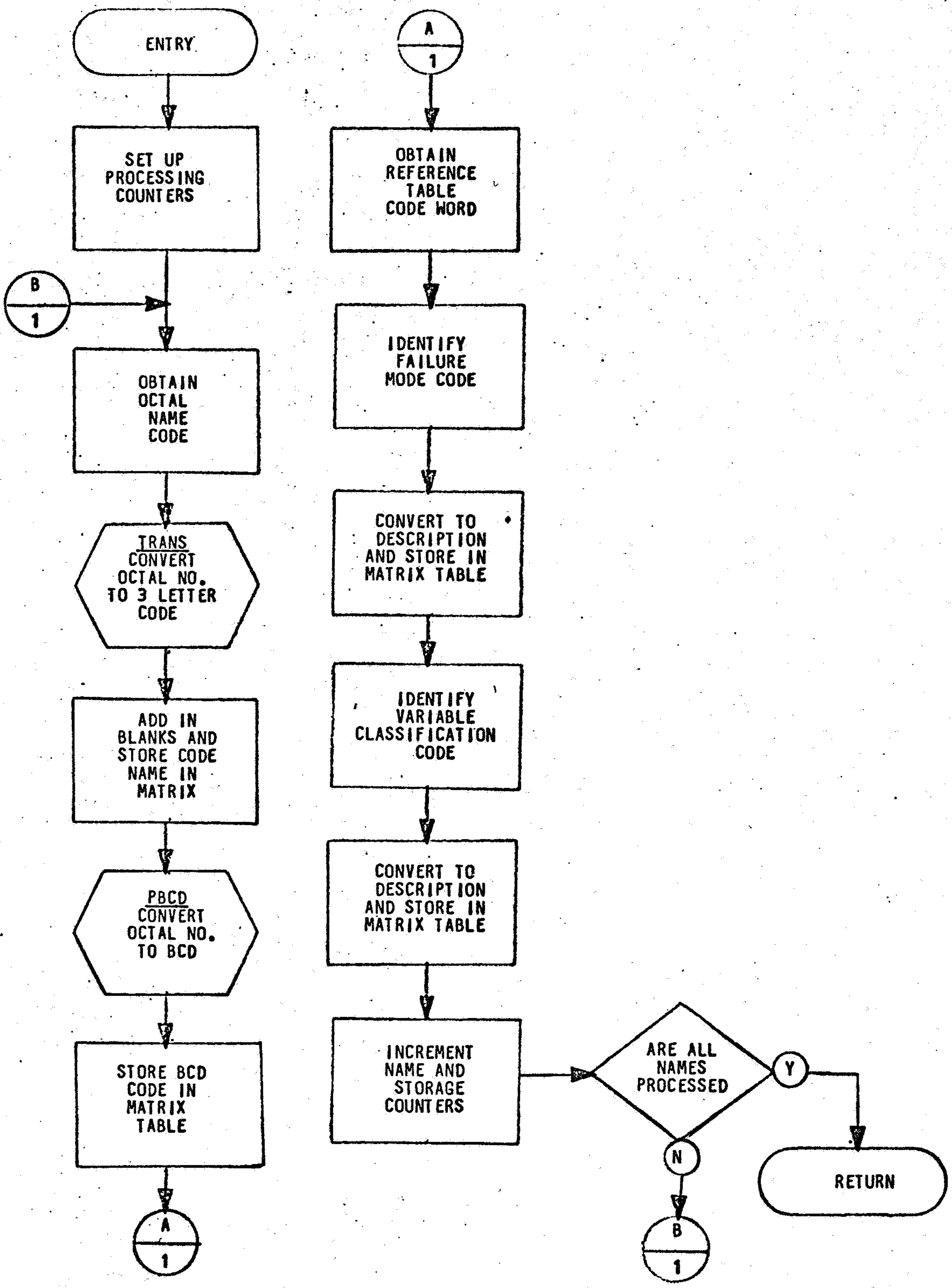

ENTRY
SET UP PROCESSING COUNTERS
B 1
OBTAIN OCTAL NAME CODE
TRANS CONVERT OCTAL NO. TO 3 LETTER CODE
ADD IN BLANKS AND STORE CODE NAME IN MATRIX
PBCD CONVERT OCTAL NO. TO BCD
STORE BCD CODE IN MATRIX TABLE
A 1
A 1
OBTAIN REFERENCE TABLE CODE WORD
IDENTIFY FAILURE MODE CODE
CONVERT TO DESCRIPTION AND STORE IN MATRIX TABLE
IDENTIFY VARIABLE CLASSIFICATION CODE
CONVERT TO DESCRIPTION AND STORE IN MATRIX TABLE
INCREMENT NAME AND STORAGE COUNTERS
ARE ALL NAMES PROCESSED
Y
N
RETURN
B 1

Figure A-4. SUBROUTINE MANIZ (1 of 1)

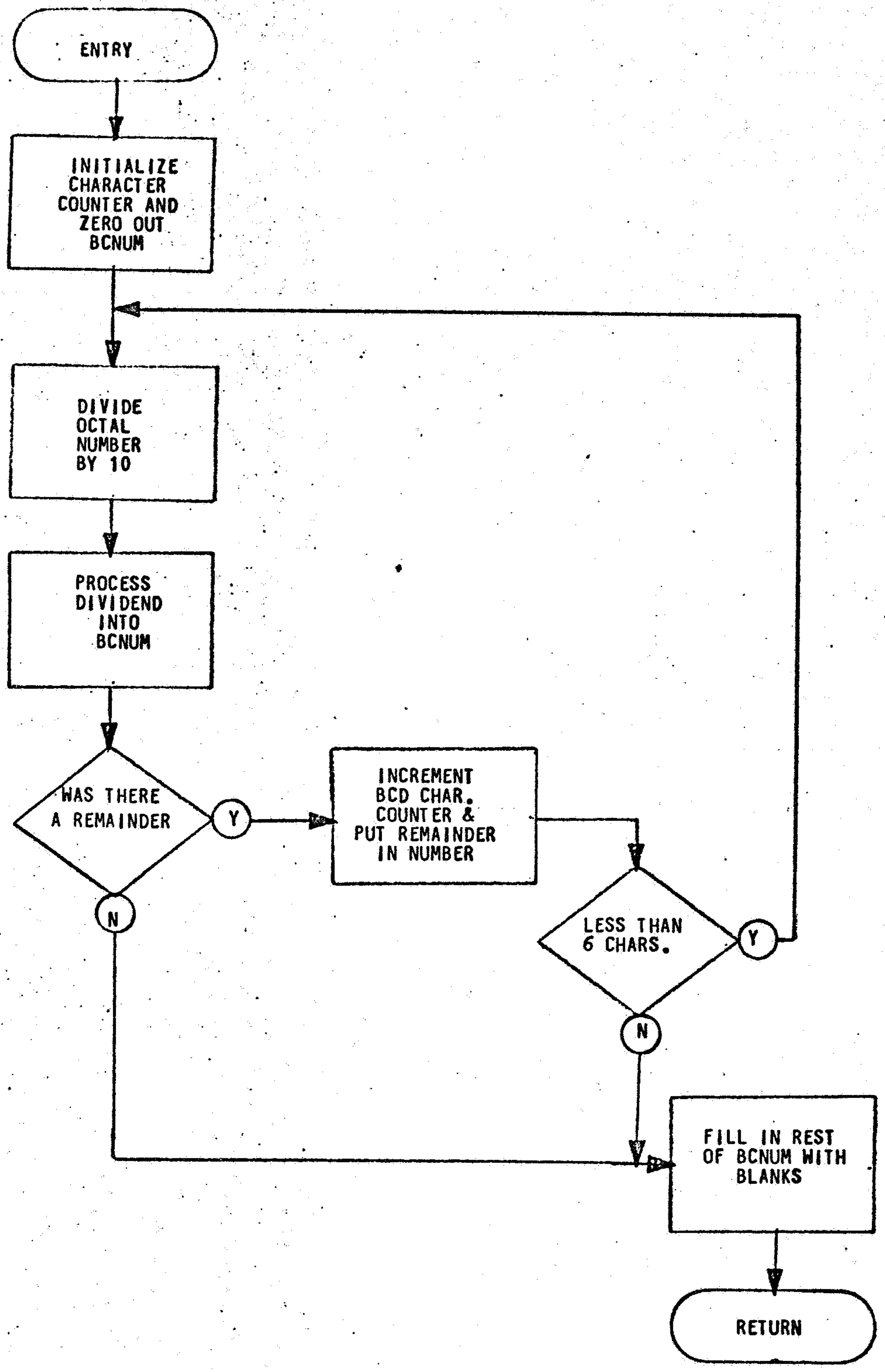

ENTRY
INITIALIZE CHARACTER COUNTER AND ZERO OUT BCNUM
DIVIDE OCTAL NUMBER BY 10
PROCESS DIVIDEND INTO BCNUM
WAS THERE A REMAINDER
Y
N
INCREMENT BCD CHAR. COUNTER & PUT REMAINDER IN NUMBER
LESS THAN 6 CHARS.
Y
N
FILL IN REST OF BCNUM WITH BLANKS
RETURN

Figure A-5. SUBROUTINE PBCD (1 of 1)

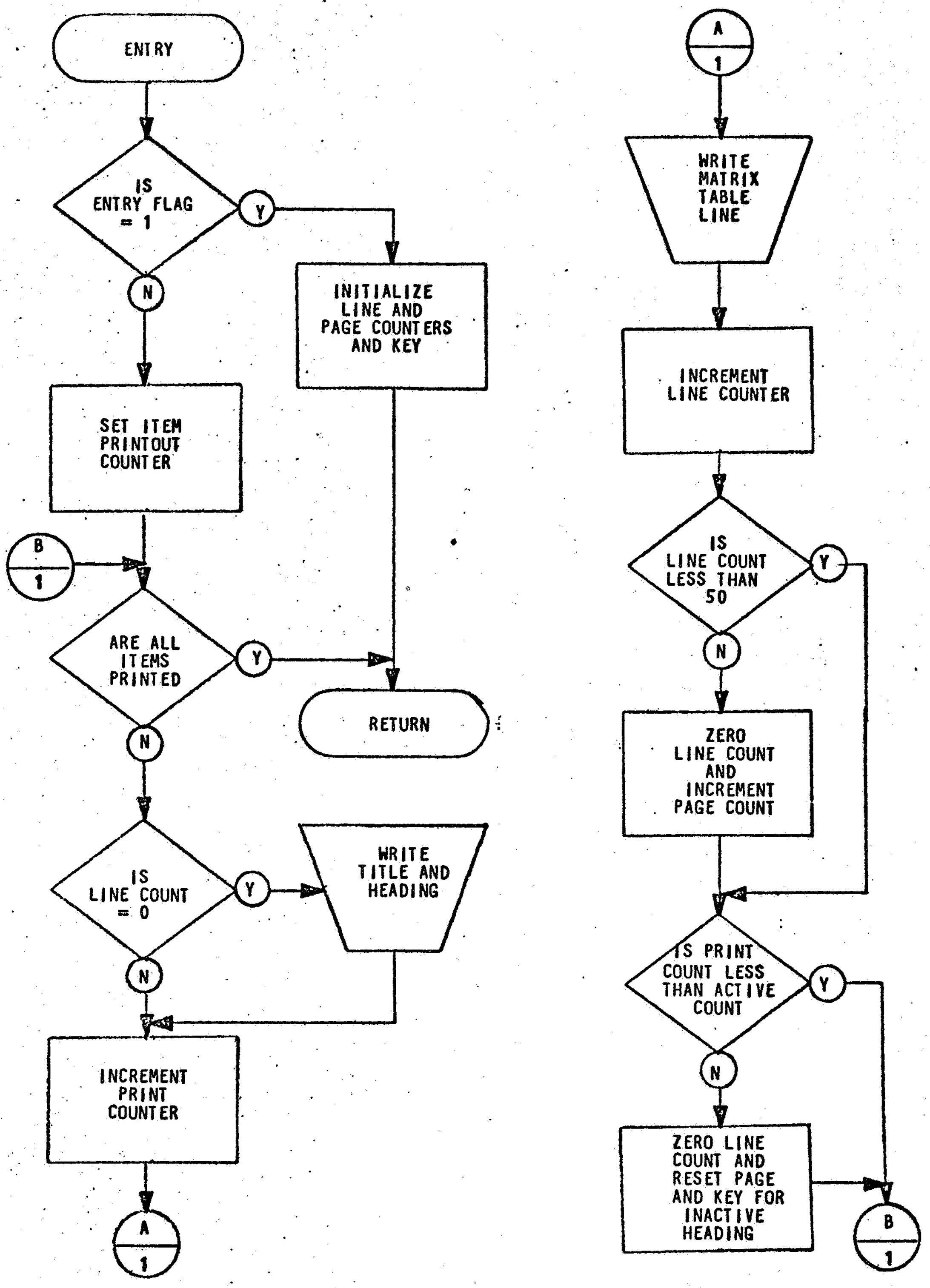

ENTRY
IS ENTRY FLAG = 1
Y
N
INITIALIZE LINE AND PAGE COUNTERS AND KEY
SET ITEM PRINTOUT COUNTER
B 1
ARE ALL ITEMS PRINTED
Y
N
RETURN
IS LINE COUNT = 0
Y
N
WRITE TITLE AND HEADING
INCREMENT PRINT COUNTER
A 1
A 1
WRITE MATRIX TABLE LINE
INCREMENT LINE COUNTER
IS LINE COUNT LESS THAN 50
Y
N
ZERO LINE COUNT AND INCREMENT PAGE COUNT
IS PRINT COUNT LESS THAN ACTIVE COUNT
Y
N
ZERO LINE COUNT AND RESET PAGE AND KEY FOR INACTIVE HEADING
B 1

Figure A-6. SUBROUTINE PRINZ (1 of 1)

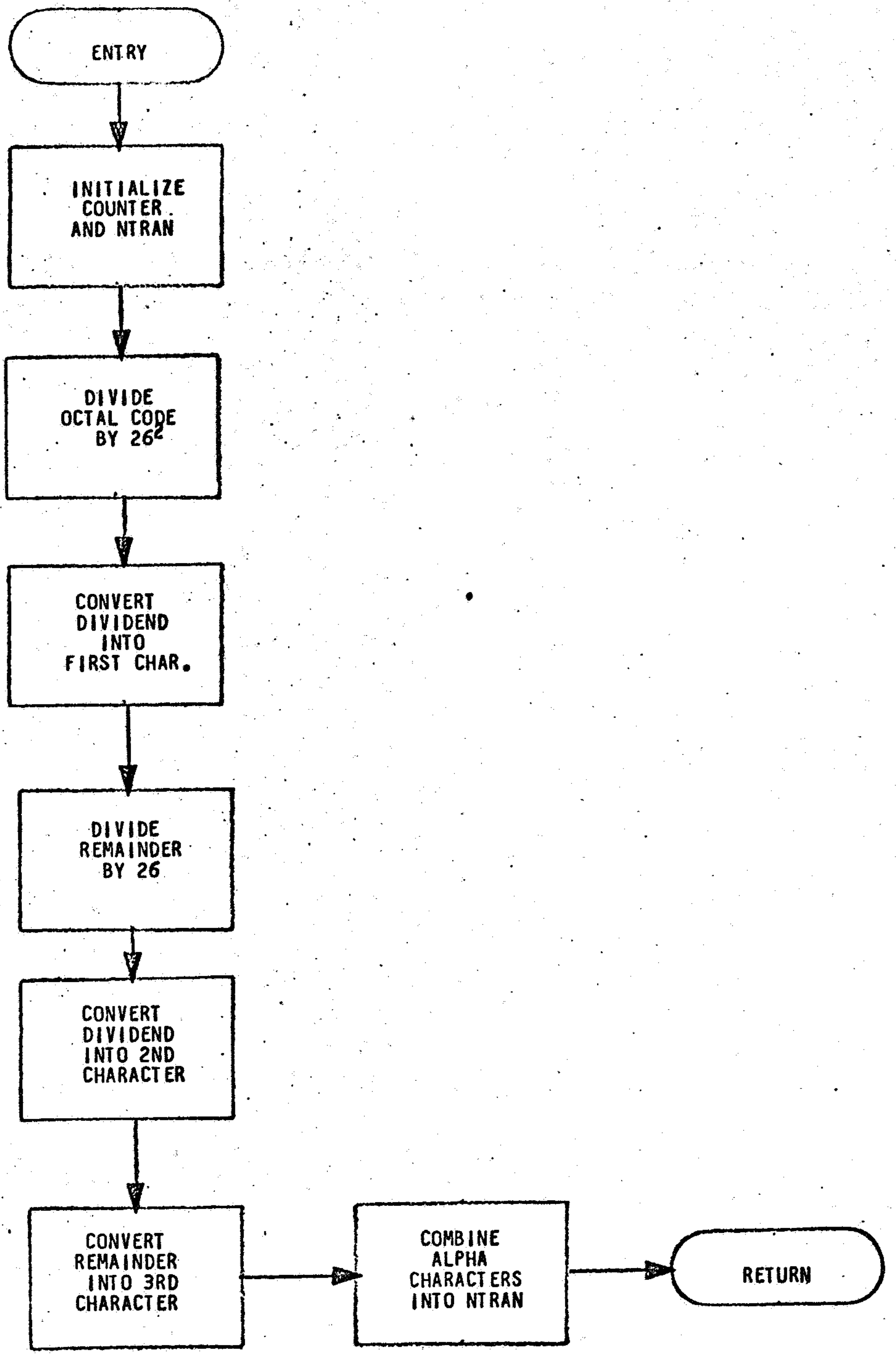

ENTRY
INITIALIZE COUNTER AND NTRAN
DIVIDE OCTAL CODE BY 26²
CONVERT DIVIDEND INTO FIRST CHAR.
DIVIDE REMAINDER BY 26
CONVERT DIVIDEND INTO 2ND CHARACTER
CONVERT REMAINDER INTO 3RD CHARACTER
COMBINE ALPHA CHARACTERS INTO NTRAN
RETURN

Figure A-7. SUBROUTINE TRANL (1 of 1)

APPENDIX B

GLOSSARY OF TERMS

1.0 INDEX OF VARIABLES

The following is an alphabetical listing of the terms used in the DTC Reference Table Program.

NAME	DESCRIPTION
ACTCT	Active name count.
APHT	Conversion table.
DM26	Conversion factor.
INACT	Inactive name count.
IPAG	Page number.
KBUF	Input buffer.
KNT	Word/record.
KREF	Reference table.
KTAB	Output array.
KTRAK	Process flag.
LCT	Line count.
NAMCT	Total name count.
NCANP	Failure type.
NITEM	Items/record.
NTRAN	Temp. conversion buffer.
NTYP	Name classification.
NWORDS	Words/record.

2.0 DEFINITIONS

NAME	DESCRIPTION
ACTCT	Contains the number of active variables in the model, and is taken from the twenty-first word of the DTC/AMA reference table prologue record. The value of ACTCT is tested in subroutines DTCINP & PRINT to provide separate printouts for the active and inactive variables.

NAME	DESCRIPTION
APHT	Table generated in subroutine TRANL to represent in BCD the twenty-six letters of the alphabet. The table is generated by three sets of DUP and VFD psuedo operators which produces a table containing octal 21 through 31, 41 through 51, and 62 through 71.
DM26	Division array for determining the APHT table address of the letters for a three letter code name. The numerical code number extracted from word 1 of KTAB is isolated and reduced by a bias of 1 for division by DM 26. The first division is by 26 squared to give the sub address in APHT of the first character for the code name. The remainder is divided by 26 for the second character, and the final remainder is the sub address for the third character.
INACT	Contains the number of inactive variables in the model, and is the value taken from the twenty-second word of the DTC/AMA reference table prologue record.
IPAG	Paging count. Value of this cell is printed at the top RH of each output page.
KBUF	A three hundred word buffer used for input storage of information read in from the DTC/AMA tape. The reference table prologue record is placed in this storage block while ACTCT, INACT and NAMCT are extracted. Each record on the tape is stored in this block while test is made for key words in the first word of records e.g. *NAMES for names record, *REFER for reference table, etc. Each names record is loaded into KBUF during the generation of the two dimensional array KTAB.
KNT	During read in of the names record KNT is incremented by one for each name processed. This count continues to be additive regardless of how many records processed. KNT is compared with the active count to separate active and inactive variable printing.

NAME	DESCRIPTION
KREF	A four thousand word storage block for the DTC/AMA tape reference table. The first word in KREF is word four of the actual reference table record. Words one through three are utilized during read in for data ID and record length. Each word in this array will contain the failure candidacy and variable classification codes which are extracted and converted in subroutine MANIP for insertion in the array KTAB.
KTAB	A 9 by 200 two dimensional array containing BCD data extracted and converted from the DTC/AMA tape in a format for printout. The array is filled for each record of names processed to the extent of the number of names in the record. This value is stored in NITEMS when the record is read. Printout is made record by record as processing of each complete names record is completed. The detailed description of data is described under Output Format.
KTRAK	The program director flag (PDF). The value or KTRAK is set prior to returns to the driver subroutine where it is tested to determine continuing flow of processing.

KTRAK = 0	Call DTCINP with PDF = 1 (1st pass)
KTRAK = 1	Call DTCINP with PDF = 2 (2nd pass)
KTRAK = 2	Call DTCINP with PDF = 3 (3rd pass)
KTRAK = 3	Call print with PDF = 1 (1st pass)
KTRAK = 4	Call DTCINP with PDF = 4 subsequent passes through subroutine to read in and process more names. Used as print check indicating active name process not completed.
KTRAK = 5	Same as 4 but signals that processing of inactive names is now underway.

NAME	DESCRIPTION	
	KTRAK = 6	Name processing completed, termination $ encountered.
	KTRAK = 7	Error in processing call ERREM with PDF = 4 (only one error print option utilized at present). Print KTRAK.
LCT	The line count of printing pages contain fifty lines of data.	
NAMCT	Total number of names derived from word 23 of the DTC/AMA reference table prologue record.	
NCAND	Seven words in storage containing BCD data for printing the failure candidacy. NCAND = BOTH, NCAND + 2 = ZEROS, NCAND +4= ONES. The remaining words are blank. When the reference data for each variable is processed, the appropriate word from NCAND is stored in KTAB (word nine) by the subroutine MANIP. Selection is accomplished by isolating bits 3, 4 & 5 of the word being processed from the reference table KREF and placing this number into XR4. NCAND is then selected as modified by XR4 which results either no printing (blanks) or one of the three words.	
NITEM	A word used for storing the value from the second word in the DTC/AMA tape record which lists the total number of items in the record. For final names record, this number will equal the number of names plus one for the terminating $.	
NTRAN	Temporary storage in subroutine TRANS used when converting from the numerical code to the three letter alpha code.	
NTYP	BCD data contained in subroutine MANIP that contains the name classifications for the corresponding codes extracted from BIT positions 18, 19 and 20 of each word of the DTC/AMA reference data in KREF. Word selection is similar to selection of NCAND.	

NAME	DESCRIPTION
NWORDS	From the third word for storing the value of the names records from the DTC/AMA tape. It provides the number of words to be read from the record. NWORDS are read from the record and stored in KBUF.

7

REFERENCES

1. "Automatic Malfunction Analysis by Discrete Network Simulation," Convair division of General Dynamics, (Report) Appendix B 60C DDF66-007, October 1966. Also see Volume Two "Simulation of Selected Discrete Networks" (Report), NASA-CR-6910.